Copy No. 16

SC5525.FR

DISTRIBUTED DECISION MAKING IN A DYNAMIC NETWORK ENVIRONMENT

FINAL REPORT FOR THE PERIOD
September 1, 1987 through September 30, 1989

CONTRACT NO. N00014-87-C-0703

Prepared for

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000
Attn: Rabinder N. Madan

A. Sastry
Principal Investigator

JANUARY 1990

Approved for public release; distribution unlimited

Rockwell International
Science Center

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SC5525.FR	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Rockwell International Science Center 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1049 Camino Dos Rios Thousand Oaks, CA 91360	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy St. Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-87-C-0703
8c. ADDRESS (City, State, and ZIP Code) 800 North Quincy St. Arlington, VA 22217-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Distributed Decision Making in a Dynamic Network Environment

12. PERSONAL AUTHOR(S) Sastry, A.R. (Principal Investigator), Baker, J.E., Clare, L.P., Fehling, M.R., Lippitz, M.J., and Richardson, J.M.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 9-1-87 TO 9-30-89	1990 January	144

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Distributed Tactical Decision Making, Value-Based Scheduling Protocols, Time-Critical Traffic, Multi-Sensor Data Fusion, Conditional Fokker-Planck Methodology, Resource-Constrained Problem Solving, Dynamic Organizational Rationality, Schemer

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This Report is concerned with distributed tactical decision making (DTDM) in a command and control system using geographically dispersed communication networks. We carried out the following investigations with the underlying motivation to model an integrated decision making environment that captures the essential real-time interactions among communications, data fusion, and decision making:

(1) Value-Based Protocols: A multiplexing protocol called Maximum Utility Scheduling For Transmission (MUST) is derived for time-critical traffic, in which messages with 'value' functions that decay with time are scheduled at a node in a dynamic manner so as to maximize the total value of all received messages. The performance of the MUST protocol is studied through the use of a Markov chain model and through simulations. Results are presented that indicate that MUST's performance is superior

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☒ DTIC USERS	
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Rabinder N. Madan	22b. TELEPHONE (Include Area Code) (202) 696-4216 / 22c. OFFICE SYMBOL ONR-1114

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

19. ABSTRACT (Continued)

to conventional protocols, particularly when traffic arrival statistics are varying or unknown, and loads are high.

(2) Multisensor Data Fusion: An essential aspect of data fusion is the estimation of the state of knowledge (i.e., the conditional probability density in state space) based upon the outputs of a dispersed set of sensors. An efficient way of handling the estimation process is to transmit bandwidth limited signals from each sensor station to a central fusion station (or several such stations) where the estimation of the most comprehensive state of knowledge is completed. Our work, based on conditional Fokker-Planck (CFP) methodology, provides a rigorous conceptual basis for accomplishing the above fusion operation efficiently, but in a manner that is exactly equivalent to the state of knowledge obtained directly from all available raw data, minimizing the need for frequent transmissions.

(3) Dynamic Organizational Rationality and Group Effectiveness: In the context of command and control, we view organizations as problem solving systems, where "problem solving" is understood to be the process of reasoning about and taking rational action to achieve some objective. Beginning with a set of primitive notions of group structure and evolution, we define concepts of rational agency, organizational intelligence, and autonomy. We call the structural point of view implied by these new definitions as 'The Organization Metaphor', and developed its functional implications via a notion that we call "agreements". Taken together, these concepts represent a theory of Dynamic Organizational Rationality. We then describe Schemer, a computational model of resource-constrained problem-solving agents and discuss how Schemer operationalizes and provides a formalizable framework within which our notions of Dynamic Organizational Rationality may be explored.

DISTRIBUTED DECISION MAKING IN A DYNAMIC NETWORK ENVIRONMENT

TABLE OF CONTENTS

PREFACE

This Report is concerned with distributed tactical decision making (DTDM) in a command and control (C^2) system using geographically dispersed communication networks. During 1987-89, under the Office of Naval Research (ONR) Contract N-00014-87-C-0703, we investigated the issues in mission-related, real-time distributed decision making considering an integrated environment in which the communication networks and protocols, sensor control and data fusion techniques, and the distributed decision making (DDM) methodology interact to accomplish the objectives of a mission.

The investigators would like to thank Dr. R.N. Madan and Dr. W.S. Vaughan, Jr. of the Office of Naval Research for their enthusiastic support of this research and interest in the subject matter. They also like to thank a number of their colleagues at the Rockwell International Science Center and other researchers in the field, too large a list to be mentioned here, for their comments and helpful discussions.

DISTRIBUTED DECISION MAKING IN A DYNAMIC NETWORK ENVIRONMENT

SC5525.FR

Executive Summary

A major concern of the Distributed Tactical Decision Making (DTDM) program of the Office of Naval Research (ONR) has been to bring coherence to various *ad hoc* studies of organizational dynamics by facilitating collaboration among researchers developing methods for evaluating alternate command structures, communication protocols, and other command and control (C^2) subproblems. In view of this, we took a broad view of research in the field and attempted to initiate groundwork for a unified framework that would help reveal relationships among the subproblems that we initially focussed on.

An example of such a large scale problem in the DTDM realm would be the evaluation of the trade-off between the efficiency of a rigid hierarchical organization in completing well-defined tasks versus the adaptability and throughput advantages of an organization in which decision making authority is distributed. Discussions at the 1988 annual coordinating meeting regarding the advantages and disadvantages of the current command hierarchy in which different functional officers report to a Combined Warfare Commander concerned precisely this trade-off.

More specifically, the following issues have been investigated with the underlying motivation to modeling an integrated decision making environment capturing the essential real time interactions among communications, data fusion, and decision making: (1) Methods of handling time-critical traffic in which messages with 'value' functions that decay with time are processed in the network in a dynamic manner so as to maximize the total value of all received messages, (2) Development of multi-sensor data fusion techniques that minimize the amount of raw data needed to be transmitted in tactical situations, and (3) Representation of a dynamic organizational framework for measuring group effectiveness for applications to command and control. The results are briefly described below.

(i) Value-based protocols for time-critical traffic: The concept of message "value" is used as a means for measuring network performance. For each message, a "value function" is defined that represents its value to the intended recipient as a function of the total delay in transporting it. These value functions not only incorporate the usual notions of relative priority (i.e., importance) between different messages, but also capture the time-critical characteristics that arise in tactical applications. Value-based network operation could permeate all layers of communications operation, provided messages are tagged with their value functions. We concentrated in this work on a single node that multiplexes messages for a single transmitter, and defined a value-based multiplexing protocol called Maximum Utility Scheduling for Transmission (MUST). The performance of the MUST protocol is studied through the use of a Markov chain model and through simulations. Results are presented that indicate that MUST's performance is superior to conventional protocols, particularly when traffic arrival statistics are varying or unknown, and loads are high. Both nonpreemptive and preemptive repeat disciplines are considered. Some possible areas for extending this research are suggested.

(ii) Data Fusion and Sensor Control: While the previous task considered efficient handling of time-critical data in a network, development of efficient real-time data fusion and

sensor control techniques that minimize the need for frequent transmissions is considered in this task. This is particularly important in tactical situations in which large masses of data can be created in a short time. An essential aspect of data fusion is the estimation of the state of knowledge (i.e., the conditional probability density in state space) based upon the outputs of a dispersed set of sensors. The current state of knowledge is evolved according to a dynamical model and is continually or intermittently conditioned on all past measurements. An efficient way of handling the estimation process is to equip each sensor station with a certain signal processing capability and then to transmit bandwidth limited signals to a central fusion station (or several such stations) where the estimation of the most comprehensive state of knowledge is completed. Our work provides a rigorous conceptual basis for accomplishing the above fusion operation efficiently, but in a manner that is exactly equivalent to the state of knowledge obtained directly from all available raw data. The rigorous fusion process is possible only in the case of deterministic dynamical models but with random initial conditions. Two versions of the conditional Fokker-Planck (CFP) methodology giving the evolution of the state of knowledge are considered: (a) a single CFP equation based upon all available sensor data and (b) a family of CFP equations each of which is based upon the sensor data available at the station with which it is associated. A simple fusion formula that relates the solution of (a) to the solutions of (b) and to the unconditional state of knowledge is derived. Practical problems of near-optimal implementation and their impact on the design of the communication system are also discussed.

(iii) Dynamic Organizational Rationality and Group Effectiveness: Beginning with a set of primitive notions of group structure and evolution, we define concepts of rational agency, organizational intelligence, and autonomy. These definitions, linked to measures of performance that follow naturally from models of perfect rationality, support the development of a normative functional theory of organizations as problem solving systems, where "problem solving" is understood to be the process of reasoning about and taking rational action to achieve some objective. The structural point of view implied by these new definitions is what we are calling The 'Organization Metaphor.' The functional implications of The Organization Metaphor are developed via a notion we call "agreements." Taken together, these concepts represent a theory of Dynamic Organizational Rationality, with which we confront some important and long-standing issues in conventional Organization Theory. We then describe Schemer, a computational model of resource-constrained problem-solving agents and discuss how Schemer operationalizes and provides a formalizable framework within which our notions of Dynamic Organizational Rationality may be explored. We conclude with a discussion of our theory and its technical implications for modeling and analyzing intelligent activity in distributed systems, particularly C^2 systems.

We believe that the above results on component elements are very useful in developing a comprehensive model for the integrated DTDM environment. We plan to incorporate these results in the future in Schemer, a computational model mentioned above that is being developed under our IR&D, to investigate the possible tradeoffs in an integrated environment. In the meantime, ONR's DTDM program (in which we participated in the last two years) has come to a close (hopefully temporarily). We look forward to having an opportunity to continue this work if ONR resumes the DTDM program.

1. INTRODUCTION

Successful operation of a distributed decision making (DDM) system requires

- Intelligent, adaptive management of communications within the system,
- Distributed information management and sensor control
- Distributed, cooperative, partitioning and sharing of various operations by human agents in the system, and
- Distributed decision making methodology and organization that coordinate individual and collective decisions and generate consequent actions.

To some extent, these areas of concern have been addressed in the literature separately and tools and methods have been developed largely in the isolated context of each area.

During 1987-89, under the Office of Naval Research (ONR) Contract N-00014-87-C-0703, we investigated the issues in mission-related, real-time distributed decision making in which the communication networks and protocols, sensor control and data fusion techniques, and the distributed decision making (DDM) methodology interact to accomplish the objectives of a mission. In this report, we first describe the integrated DDM problem and identify three interacting and yet distinct component areas that should work together for effective decision making. We then present briefly the approach to the problems chosen in each of the individual areas and the results obtained. More detailed description of the problem, related work in the literature, approach, results, and references are given for each of the three tasks in the corresponding Appendixes I-III.

On a long term basis, we adopted a more integrated approach to these problems by formulating a rigorous conceptual framework that explicitly identifies the functional relationships among the elements that influence the readiness and responsiveness of the total system. Figure 1 gives a schematic representation of the interaction among decision making, data collection, and communication network elements. The common goal of all the elements is to support the execution of a mission to realize the mission requirements. The decision making is carried out by cooperating agents based on available data on mission status. These decisions pertain to various aspects of mission execution, including sensor control and network management. Since the communication network is the means of not only assessing the mission's status but also transporting battle management decisions, the network also needs to be closely monitored and maintained so as to be able to support the overall mission. One of the key issues is that while demands may not be made on a network that cannot be met, it is also important to fully exploit the network's agility for real-time management of a mission.

This conceptual integrated framework functions as the foundation for detailed investigations on some key issues related to the design and operation of DDM systems. Figure 2 shows a layered structure that broadly groups the interdependent issues into three areas for the sake of defining focussed effort; (i) cooperative decision making, (ii) data fusion and sensor control, and (iii) networks with time-critical traffic. Each layer will specify the 'requirements' to the layer below to meet the needs of mission related decision making. Joel Lawson* identified several steps involved in decision making, such as sensing, assess-

* J.L. Lawson, Jr., *Program Review - Distributed Tactical Decision Making Program*, November 1986

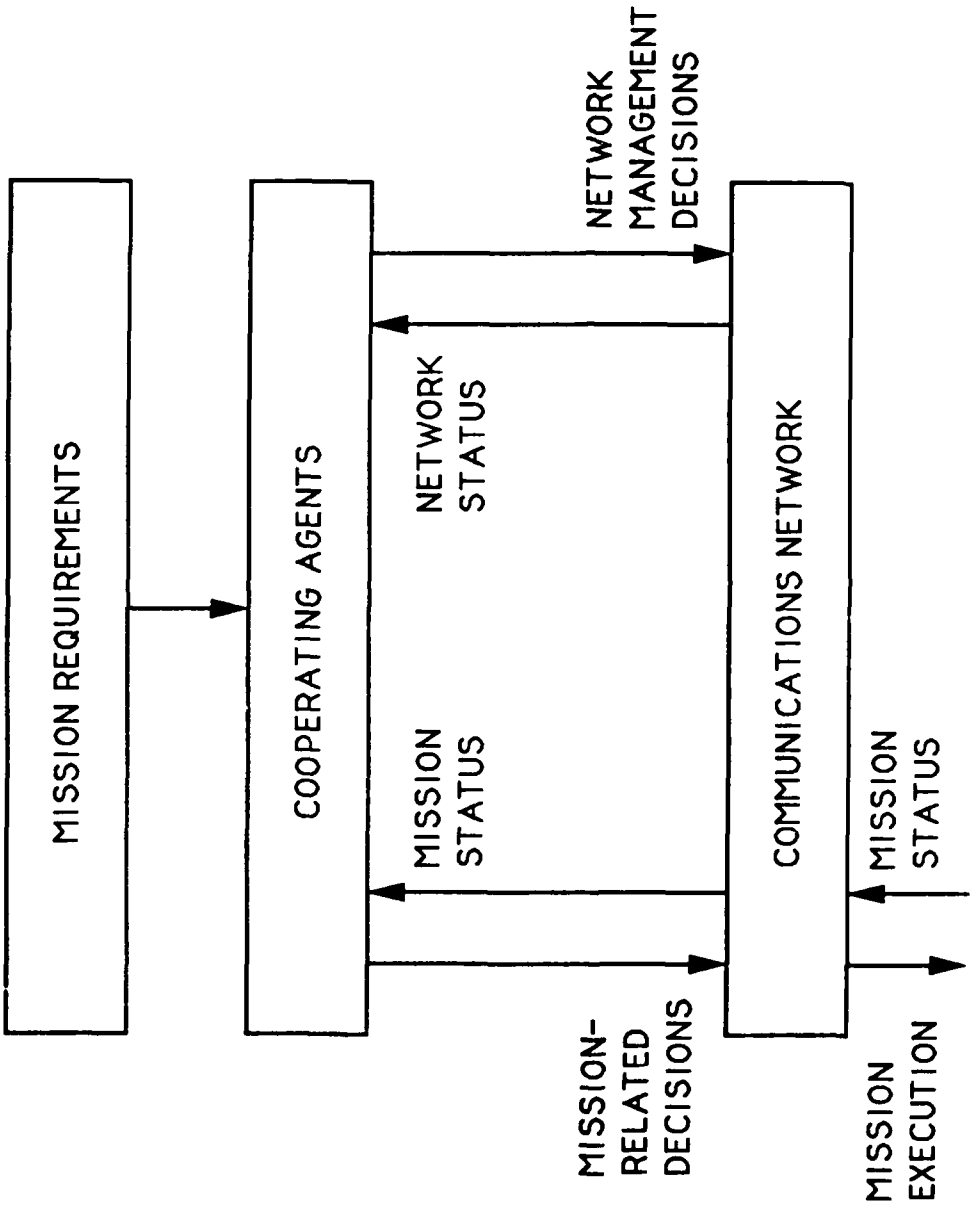


Figure 1. Interaction among decision making, data collection, and network elements

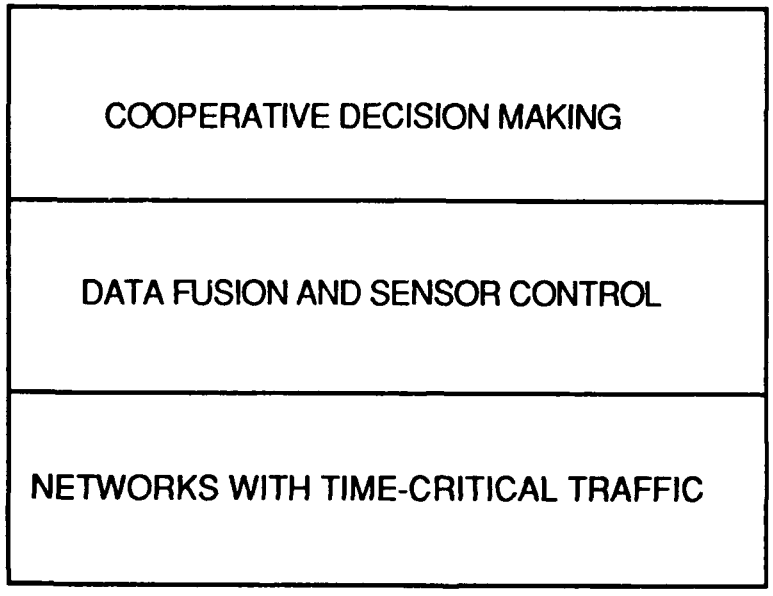


Figure 2. Distributed tactical decision making in a dynamic network environment: Functional layers

ment, evaluation of options, choice of action, and execution. Using such stages in decision making with the required granularity, appropriate requirements to be met at each layer can be generated such as, for example, the real-time response needs to be met by the network.

More specifically, the following issues are considered in our work:

(1) Value-based Protocols for Time-Critical Traffic: We consider tactical situations in which we associate with each message a 'value' function that decays with time in a predetermined manner. The value at any given time may represent an inherent level of mission-related usefulness of the message and the priority associated with it. Messages whose values decay to zero may be removed from the network immediately. However, messages that have finite residual value need to be processed in a dynamic manner so as to maximize the total value of all received messages. We have developed appropriate queueing protocols for handling such time-critical traffic and evaluated them through analysis and simulation.

(2) Multisensor Data Fusion with Minimal Transmission: An essential element of a distributed decision making methodology is the estimation of the state of knowledge (i.e., the conditional probability density in state space) based upon the outputs of a dispersed set of sensors. The current state of knowledge is evolved according to a dynamical model and is continually or intermittently conditioned on all past measurements. An efficient way of handling the estimation process is to equip each sensor station with a certain signal processing capability and then to transmit bandwidth limited signals to a central fusion station (or several such stations) where the estimation of the most comprehensive state of knowledge is completed. This task provides a rigorous conceptual basis for accomplishing the above fusion operation efficiently with reduced need for data transmissions, but in a manner that is exactly equivalent to the state of knowledge obtained directly from all available raw data, in the case of deterministic dynamical models but with random initial conditions.

(3) Cooperative Decision Environment: In the context of C^2 issues, we view organizations as problem solving systems, where "problem solving" is understood to be a resource-constrained process of reasoning about and taking action towards an objective. Beginning with a set of primitive notions of group structure and evolution, we define concepts of rational agency, organizational intelligence, and autonomy. These definitions, linked to measures of performance that follow naturally from models of perfect rationality, support the development of a normative functional theory of organizations as problem solving systems. The structural point of view implied by these new definitions is what we are calling The 'Organization Metaphor.' The functional implications of The Organization Metaphor are developed via a notion we call "agreements." Taken together, these concepts represent a theory of Dynamic Organizational Rationality, with which we confront some important and long-standing issues in conventional Organization Theory. We then describe Schemer, a computational model of resource-constrained problem-solving agents and discuss how Schemer operationalizes and provides a formalizable framework within which our notions of Dynamic Organizational Rationality may be explored.

2. OBJECTIVES

The objectives of the work are to develop a comprehensive understanding of the issues that strongly influence the design of real time, tactical, distributed decision making and to specify appropriate measures of effectiveness of the total system. In this context, our near term objective is to conduct detailed investigations in communications, computing, data fusion, sensor control, and cooperative decision making by taking into account their mutual interactions. Our long term goal is to develop an integrated framework and a computational model that would serve as vehicles to evaluate various system tradeoffs under realistic scenarios. Using these models, we also seek to identify any incompatible situations clearly, i.e., inability of the network to respond to the decision demands or deficiencies in the decision making process that prevent exploitation of the full potential of the resources and agility of the network. The results of these investigations are expected to contribute to the methodology in designing the decision making systems and to measure their effectiveness.

3. VALUE-BASED PROTOCOLS

Consider a user of communications network who wishes to have a message delivered. The level of service provided to a communication network user is typically described in terms of the number of messages that are delivered correctly, and the delays involved. Transcending this is the actual utility or value of the message to the recipient and/or sender. Normally, some messages are more valuable than others, and message values may decrease as their transport delays increase. Many prioritized message-handling schemes have been proposed that indirectly account for these factors by treating different message classes differently. However, these schemes typically require detailed knowledge of the traffic statistics and delay requirements and are not robust to changes in either. In this paper, we propose a general scheme that directly uses message values and their time dependencies for the scheduling of transmissions. We assume that values can be calculated for delivered messages, and that values for different users can be compared. Our initial objective is maximize the *total* value delivered. This implicitly accounts for message delays and losses.

We consider a generalization of the usual performance metrics of delay and throughput by specifying "value functions" that represent the utility of a message to the recipient as a function of the total delay in its transport. Further, we consider the operation of a network in which each message is tagged with its value function and these tags are used to dynamically allocate the communications resources in order to maximize utility. For simplicity, we concentrate on the value-based operation of a single statistical multiplexer. Messages whose values would be zero can be discarded; in this sense, flow control (congestion control) is being performed. Note that such discards improve performance from the perspective of the network provider, since communications resources are not expended needlessly.

We present (1) a formal definition of message value by which performance is measured, (2) the synthesis of a realizable value- based protocol called Maximum Utility Scheduling for Transmission (MUST), and (3) numerical examples that indicate that, in certain cases, substantial gains can be achieved by the use of value-based operation in comparison to

conventional protocols. In particular, the MUST protocol is very effective when traffic statistics are varying or unknown, when loads are high, and when the system is transmitting a mixture of low priority messages with short deadlines and high priority messages with long deadlines. We feel that the requirements of military and factory communication networks are likely to correspond exactly to the situations where value-based operation promises the greatest advantages.

Our goal in this initial effort is primarily to determine whether value-based statistical multiplexing can offer substantial performance enhancement. The conclusion to be drawn is that value-based operation offers performance at least as good as any conventional protocol under any scenario, and can yield substantially improved performance than any conventional protocol in some scenarios. There are many directions for extending the research in value-based network operation, and our work indicates that pursuance is warranted.

The approach and the results obtained so far are briefly described below. Appendix I gives a detailed treatment of the concept of value-based operation and describes the results of our theoretical and simulation work.

3.1 System Model

The basic concept of handling messages in a communication system based upon their utility is illustrated in the context of statistical multiplexing, which consists of the following elements. There is a single transmitter (single channel) with a single infinite capacity buffer (when we later consider a Markov chain model of the system, we will assume that the buffer has limited capacity.) Messages arrive at random times, causing a queue to occasionally form. The message lengths can be variable, but are assumed to be known at the time of arrival, so that it is known how long it will take to transmit each message. Each message is tagged with a deterministic *value function,* that indicates the value (or utility) that the message has to the receiver as a function of the time between the message arrival at the transmitter and the completion of the message transmission. All messages are assumed to be autonomous in the sense that their value to the receiver depends only on the delay in transport of that message, and does not depend on the delivery characteristics of other messages.

The queueing discipline chooses the next message in the buffer that is to be transmitted. Messages may also be removed from the buffer and discarded. In addition, a message being transmitted can be preempted, and either placed back in the buffer or discarded. A preempted message must be completely retransmitted, i.e., preemptive repeat is assumed. We seek to synthesize a scheduling policy, and possible preemption procedure, so that the rate of received message utility is maximum.

3.2 The MUST Protocol

We propose the following procedure, referred to as the Maximum Utility Scheduling for Transmission (MUST) protocol.

Messages arrive at a multiplexer and wait for transmission. Their arrival times and lengths are governed by arbitrary and possibly unknown stochastic processes. Each arriving message has an associated value function, that gives the value of the message to system as a function of its delay (time in system.) Whenever a new message arrives, or a transmission is completed, the scheduler considers the entire collection of waiting messages

(plus the message in service if preemption is allowed) and considers each possible transmission sequence (i.e., each possible permutation of the waiting messages,) *assuming that no further messages arrive in the system.* Since message lengths are known, the scheduler can determine when the transmission of each message in a sequence will occur, and, thus, the delay and corresponding value for each message. The scheduler determines the value-maximizing schedule and transmits according to it. If a new message arrives, the tentative schedule is recomputed. The scheduler knows when messages reach a value of zero and discards them. If a message is preempted, the entire message will have to be repeated. When the queue size is large due to frequent arrivals of new messages, the computational burden required by MUST may be reduced by forming schedules that fit within a finite time horizon, h.

The MUST protocol is not necessarily optimal for any particular arrival process, but can generally be expected to yield excellent performance compared to other nonanticipatory policies (i.e., policies that do not incorporate a priori knowledge of the arrival process.) It is realizable, although it can require considerable computation if the buffer size X is large (about ten or more).

We used two complementary performance evaluation approaches: simulation and analytical modeling. The simulation model allows general value functions, continuous time operation, and preemptive or nonpreemptive service. For reasons of tractability, the analytical model is limited to a slotted system with a fixed number of message classes and nonpreemptive service.

3.3 The Simulation Model

The simulation of the baseline protocol has the same difficulties as would an actual implementation in terms of the computational burden of comparing all possible schedules at each arrival. Our simulation of the baseline protocol currently employs a brute force procedure that exhaustively enumerates all possible cases, and chooses the best one. To obtain a quantitative feeling for the relative performance of the MUST scheduling policy, we have conducted extensive computer simulations and compared it with the conventional policies of First-In-First-Out (FIFO), Last-In-First-Out (LIFO), and Head-of-Line (HOL), also known as strict or fixed priority. Furthermore, we have considered two types of variations that can occur: whether or not preemption of the ongoing transmission is allowed, and also whether the protocol discards (without transmission) all messages that would have zero value by the time its transmission completed. The specific protocols simulated were: (1) FIFO, (2) FIFO with discarding of valueless messages, (3) nonpreemptive LIFO, (4) nonpreemptive LIFO with discarding, (5) nonpreemptive HOL, (6) nonpreemptive HOL with discarding, (7) preemptive repeat HOL, (8) preemptive repeat HOL with discarding, (9) nonpreemptive MUST with possibly finite horizon h, and (10) preemptive repeat MUST with possibly finite horizon.

The performance will depend on the arrival process. Clearly, the largest differences between MUST and any other protocol will occur when queue sizes at least occasionally become moderately large, since otherwise there is essentially no choice to be made in scheduling. Large queues result if either the arrivals occur in bursts or the load (arrival rate to service rate ratio) is high. We could contrive a burst arrival process that makes MUST look arbitrarily better than other disciplines, but we instead choose the more common

and realistic assumption of Poisson arrival processes. We generally selected the aggregate load to be 100%, so that sizable queues will occur. The simulation runs were made for a simulated time interval of one second, with a channel transmission rate of 1Mb/s and packet lengths nominally of 1000 bits. For all but the FIFO case without discarding, the results are representative of steady state conditions. The MUST protocol allows complete freedom in selecting value functions associated with the messages. However, in the numerical examples shown in this paper, we limit ourselves to only two simple forms for the value functions: hard deadline and linear decay. Each such function is defined by two parameters, the initial value and the deadline.

3.3.1 Simulation Results

All of the results presented are limited to only two types of traffic sources, where all messages generated by a given "source" have the same value functions. Figure 3 illustrates the performance of the various protocols when the two types of value functions are linear as shown and all messages have the same length of 1000 bits (this is taken to be unit length). Performance is compared as a function of the deadline for the messages. The MUST protocol with preemption is consistently best, although for this case the HOL discipline with preemption and discarding of valueless messages is almost as good. In fact, we generally found that discarding of valueless messages for any protocol results in considerable performance gains. (Of course, this requires the messages to be tagged with their deadlines.) Similar results have been obtained (see Appendix I) comparing the various protocols when the value functions for all messages are the as shown same but the two sources differ in terms of the message lengths. We take the ratio of the message lengths to be unity, and plot the performance as one of the message lengths is enlarged. Again, MUST performs at least as well as all other protocols over the entire range. LIFO with discarding performs well when there is little message length disparity and HOL with discarding performs well as the disparity increases. If the the message length disparity is time-varying then MUST's performance will be superior to the other protocols. In general, MUST is robust in that the traffic characteristics need not be known beforehand and may vary over a wide range.

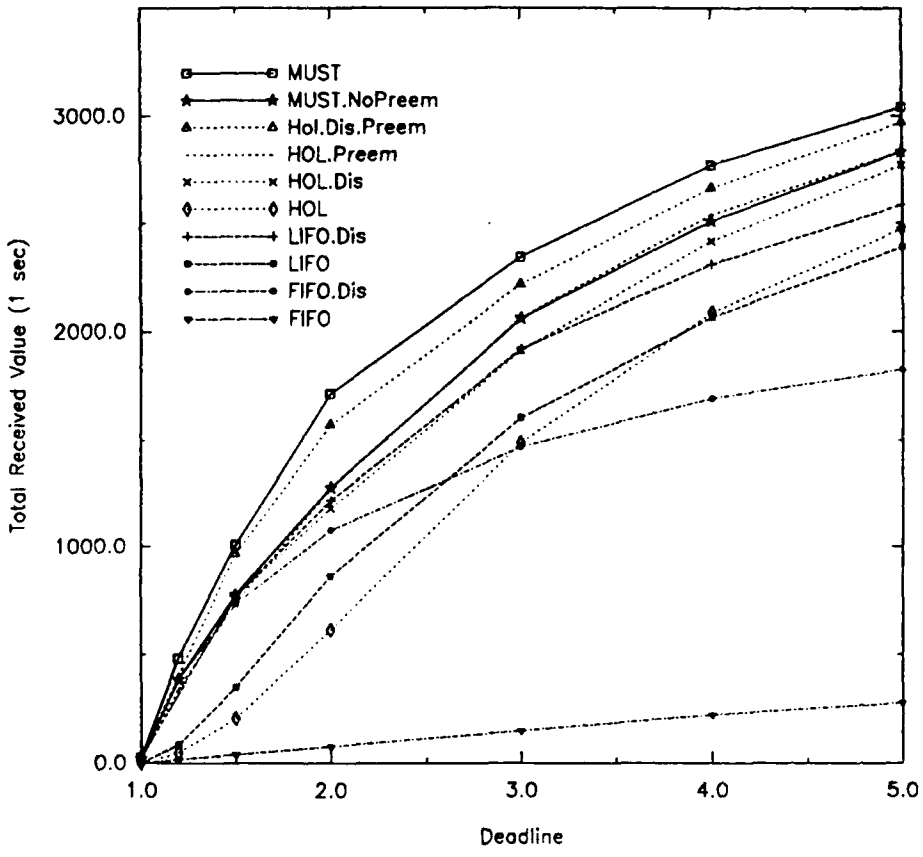
The performance of MUST as a function of the horizon h has also been examined. Two cases are considered, with two traffic source types each. In both cases, a horizon value of only about 2 message transmission times is adequate to approach optimal performance. When very small horizon times are chosen, many ties occur among the schedules, and the heuristic for breaking ties becomes important. In these examples we simply used a FIFO rule for breaking ties, so that the performance as h tends to 0 is equivalent to FIFO with discarding.

3.4 The Analytical Model

The system model used in the simulations of the MUST protocol is very general and not amenable to an exact analytical evaluation. Therefore, we will present a less general model that can be modeled analytically.

3.4.1 Model Features

In particular, we consider a discrete time model of a multiplexer. The time axis is *slotted*. All message transmissions begin at the start of a slot and continue for an integral



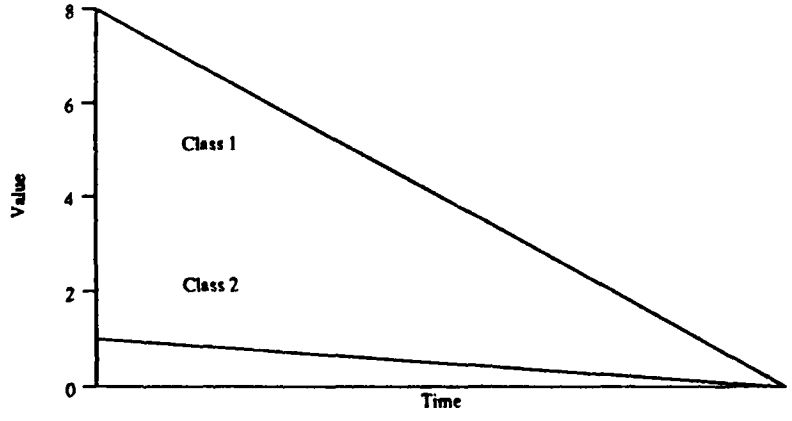


Figure 3: Comparison of protocols where value functions linearly decreases to deadline, varying deadline

number of slots. Message lengths and delays are measured in slots. The multiplexer is modeled as a single server queue. Messages arrive at the queue according to a random process. Each message has an associated value function. We restrict messages to belong to one of several classes. All messages in a class have the same length and the same value function.

The server transmits messages nonpreemptively. At the end of a message service, the server chooses another message for service according to the MUST protocol; or if there are no ready messages, the server remains idle until a slot contains arrivals. A message is then chosen for service from among those arrivals according to the MUST protocol. If a message's value function reaches zero while it is still waiting for transmission, the message is discarded. The queue size is limited for each message class. If messages need to be discarded to stay under the limit, the oldest messages are discarded and new arrivals are kept.

3.4.2 The Environment

We consider the operation of a multiplexer in a changing environment, where traffic conditions may vary rapidly, traffic sources may be inter-related, and overloads may occur. Message value functions could also depend upon the environment, but this is not explored in the current work. The random environment is modeled by a finite state, irreducible and aperiodic Markov chain By using a Markovian model of the environment, we can study the performance of the multiplexer under highly varying traffic loads and varying conditions, while retaining analytic tractability.

3.4.3 The Arrival Processes

Each message that arrives at the multiplexer belongs to one of several classes. Messages that arrive during a slot are recorded at the end of the slot. Variable message lengths could be considered by breaking a class into separate classes, each with fixed length messages. Note that the scheduler is assumed to know the lengths of the messages waiting in its queue. Many scheduling papers have assumed that message lengths are exponentially distributed, and unknown by the scheduler, which is an unrealistic assumption for many systems.

The joint distribution of the number of messages of each class that arrive in a single slot can depend upon the value of the environment variable for that slot. Conditioned upon the value of the environment, the joint distribution is independent of the past and future of the system. Thus, we have a Markov-modulated joint arrival process. Message arrivals from different classes are allowed to be correlated within a slot, and, through the state of the environment, from slot to slot.

3.4.4 Message Ages and Phases

Message values generally decrease as they age. Voice packets or control messages may have hard deadlines, a message containing position information about a moving object will become less accurate as time passes, and so on. Therefore, message phases (a generalization of message age) are tracked for use by the MUST protocol. The value function is a deterministic function of message phase at delivery and is the same for all messages of a given class. (The value could also depend on the state of the environment at delivery, but that is not considered in this paper.) The value function gives the value of a message to the system. The objective of MUST is to maximize the total value delivered by the

system. For each class, the value of a message is a nonnegative, nonincreasing function of the message's phase. More details of the analytical model are available in Appendix I.

3.4.5 Numerical Examples

The performance of the MUST protocol under a variety of conditions is investigated and compared to other scheduling disciplines. Specifically, it is compared to first in, first out [FIFO] ; last in, first out [LIFO]; shortest time to extinction [STE]; and head of line [HOL]. Figure 4 shows the results obtained from the analytical model for MUST and the other scheduling protocols.

For FIFO, LIFO, and HOL, both the case where the disciplines completely ignore the value functions, and the case where they discard messages that have reached a value of 0 are considered. The comparison to these disciplines gives a measure of the benefit achievable through the use of value functions, whether through the MUST protocol or through a simpler value-based heuristic. The various protocols were implemented by calculating the appropriate decision matrix for each protocol.

3.5 Conclusions and Extensions

We have shown that the concept of assignment of value functions for time-critical messages in a tactical context is a useful one that can be exploited in designing efficient scheduling protocols. We have developed a protocol called Maximum Utility Scheduling for Transmission (MUST) that appears to be very efficient and robust under different types of traffic conditions. Analytical formulation and derivation of closed form solutions seem to be quite complicated. The quantitative results described are based on both simulation and analytical models, and have given an excellent preview of the potential performance capabilities of MUST. In our future work, apart from developing these results further at a node level, we will consider their extension to the network level, introducing such issues as value-based routing, delay estimations, assignment of value functions in relation to mission objectives, joint value functions for correlated messages, and implementation issues and costs. Details of these investigations are given in Appendix I.

4. MULTISENSOR DATA FUSION WITH MINIMAL TRANSMISSION FOR TACTICAL SYSTEMS

Distributed decision making is a methodology with increasing importance in many applications (e.g., command of naval fleets, battle management of ground forces, anti-submarine warfare, SDI, and many others). An essential aspect of this methodology is the estimation of the state of knowledge (i.e., the conditional probability density in state space) based upon the outputs of a dispersed set of sensors. The current state of knowledge is evolved according to a dynamical model and is continually or intermittently conditioned on all past measurements. An efficient way of handling the estimation process is to equip each sensor station with a certain signal- processing capability and then to transmit bandwidth limited signals to a central fusion station (or several such stations) where the estimation of the most comprehensive state of knowledge is completed.

This work provides a rigorous conceptual basis for accomplishing the above fusion operation efficiently, but in a manner that is exactly equivalent to the state of knowledge obtained directly from all available raw data. The rigorous fusion process is possible only in the case of deterministic dynamical models but with random initial conditions. Two

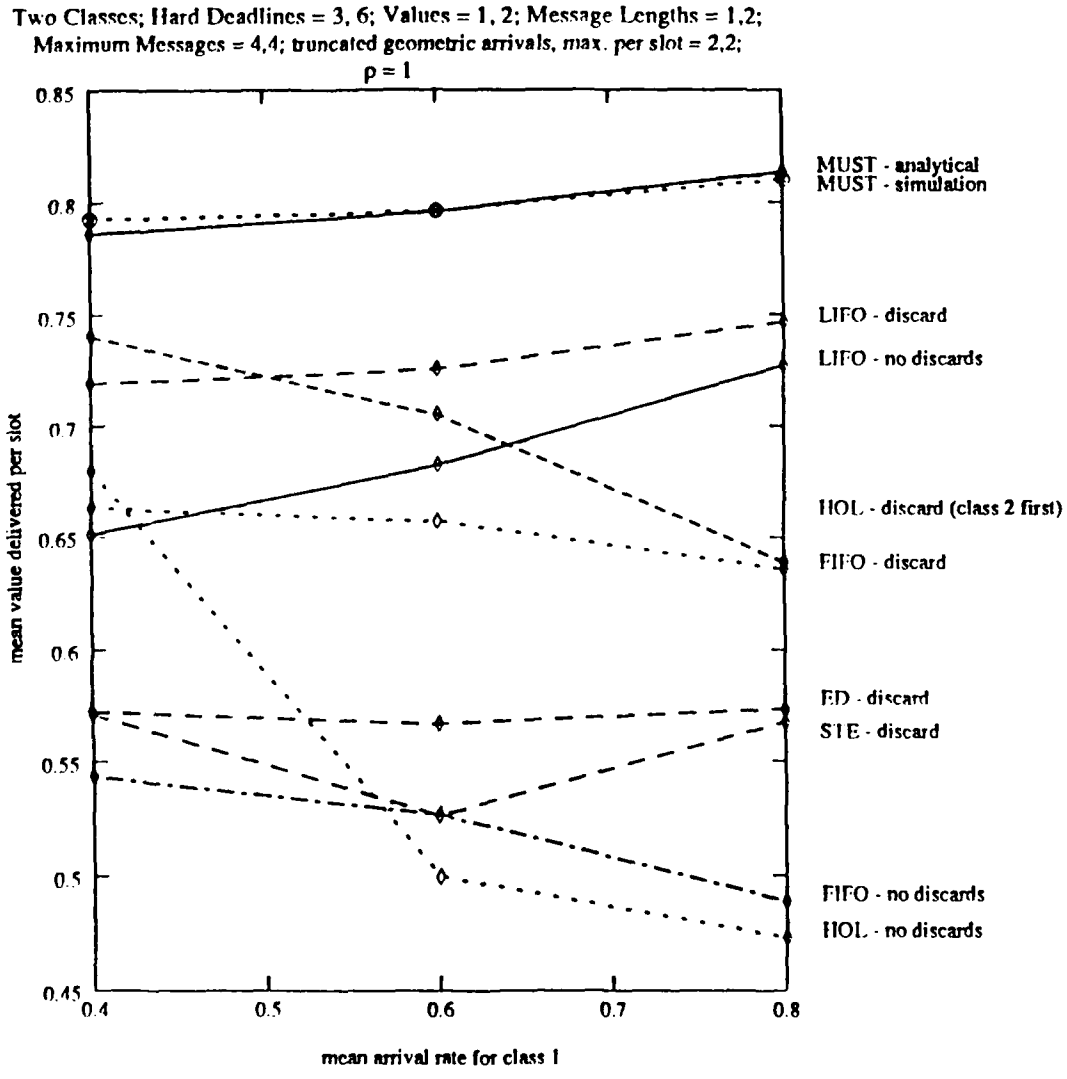


Figure 4. Analytical comparison of MUST and other scheduling protocols: Hard deadlines

versions of the conditional Fokker-Planck (CFP) methodology giving the evolution of the state of knowledge are considered: (a) a single CFP equation based upon all available sensor data and (b) a family of CFP equations each of which is based upon the sensor data available at the station with which it is associated. A simple fusion formula that relates the solution of (a) to the solutions of (b) and to the unconditional state of knowledge is derived. Practical problems of near-optimal implementation and their impact on the design of the communication system are discussed.

Let us consider the determination of the optimal estimate based upon all data and then later generalize the formulation to embrace the problem of fusion. The system of interest is assumed to be a single object (e.g., a missile in an exo-atmospheric environment or a submarine in the ocean). We make the important restrictive assumption that the object is at most subject to "parametric maneuvering," namely its dynamics (including parametric maneuvering) is described by deterministic differential equations with random initial conditions.

We accordingly assume that the dynamical and measurement models are represented by

$$x(t) = f(x(t), t), \tag{1}$$

$$y(t) = g(x(t), t) + \nu(t), \tag{2}$$

where

$x(t)$ = state of the object at time t (an m-dimensional vector) including maneuvering parameters,

$f(x(t), t)$ = an m-dimensional vector function of x(t) and t giving the rate of change of x(t) in the absence of noise,

$y(t)$ = the set of measurements at time t (an n- dimensional vector),

$g(x(t), t)$ = an n-dimensional vector function of $x(t)$ and t, and

$\nu(t)$ = an n-dimensional Gaussian random process.

This Gaussian random process has the a priori statistical properties

$$E[\nu(t)] = 0, \tag{3}$$

$$E[\nu(t)\nu(t')^T] = R_\nu \delta(t - t'), \tag{4}$$

where $R\nu$ is a constant symmetric, positive-definite nxn matrix. We further assume that $\nu(t)$ and $x(t')$ are statistically independent when $t > t'$. Now, the problem at hand involves the consideration of separate surveillance operations carried on by two sensor stations, each with its own independent sensor system. Denoting the measurement vectors for the two sensor systems by $y_1 = y_1(t)$ and $y_2 = y_2(t)$, the processing in the two stations can be represented by the two measurement models.

$$y_i(t) = g_i(x(t), t) + \nu_i(t), i = 1, 2, \tag{5}$$

where the $\nu_i(t)$ are statistically independent Gaussian random processes with

$$E[\nu_i(t)] = 0, and E[\nu_i(t)\nu_i(t')^T = R_{\nu_i}\delta(t-t'), i = 1,2. \quad (6)$$

For the sake of brevity, a quasi-rigorous discussion of the fusion problem is presented here. The measurement vectors $y_1(t)$ and $y_2(t)$ represent the measurements made at time t at sensor stations 1 and 2, respectively, and the state vector $x(t)$ gives the state of an object at time t (assuming that only one object is present). y_1^t, y_2^t and x^t denote the complete time-histories of $y_1(t), y_2(t)$, and $x(t)$ in the interval $(0,t)$. In the dynamical case the assumption of the conditional independence of y_1^t and y_2^t must be expressed in the form

$$P(y_1^t, y_2^t|x^t) = P(y_1^t|x^t)P(y_2^t|x^t). \quad (7)$$

Because of the assumption that the dynamical model is deterministic, but with random initial conditions, it follows that the state $x(t')$ at any time t' determines the entire history of x, and in particular $x(t)$ implies x^t. In this case, (7) reduces to

$$P(y_1^t, y_2^t|x(t)) = P(y_1^t|x(t))P(y_2^t|x(t)). \quad (8)$$

Some straightforward manipulations yield the relation

$$P(x(t)|y_1^t, y_2^t) = a(t)P(x(t)|y_1^t)P(x(t)|y_2^t)P(x(t))^{-1}, \quad (9)$$

where $a(t)$ is a normalization constant. To make the above results rigorous the abstract mathematical apparatus of the theory of stochastic processes must be used. However, this problem can be handled simply and rigorously within the mathematical framework of the CFP equation, as is shown in Appendix II.

The results derived are at an abstract level where proof of principle takes precedence over problems of implementation. We must now turn to consider the practical problems entailed in implementation. First of all, clearly some kind of approximate representations of the probability densities (PDs) must be devised so that approximate versions of $P(x(t)|y_1^t)$ and $P(x(t)|y_2^t)$ can be transmitted from the sensor stations 1 and 2 to a fusion station where the total CPD $P(x(t)|y_1^t, y_2^t)$ is computed using an approximate version of Eq.(9), where a is a normalization factor. If all of the CPDs are Gaussian the problem is straightforward. In the non-Gaussian case (due to nonlinearities in dynamical or measurement models, or to the presence of several objects) more than means and variances must be used to represent the CPDs (e.g., use Gram-Charlier series, Gaussian sum approximations). Returning to the question of many objects, it may be more appropriate to consider various orders of conditional mean densities in single-object state space and an associated hierarchy of equations giving their time evolution. Various approximation methodologies must be devised for reducing this formulation to computationally tractable form.

In any case, the idea that the transmission of CPDs from sensor stations to a fusion station entails less load on the communication system than the transmission of the raw data depends, of course, on two factors: (1)how often the CPDs are transmitted and (2)what kinds of approximate representations of the CPDs are used. We have assumed that these two factors can be handled such that it is distinctly advantageous to transmit approximate representations of CPDs.

Another point to consider is that in the transmission of CPDs no supplementary information is required concerning what measurements were made and when they were made. On the other hand in the transmission of raw data all supplementary information is essential.

Details of these investigations are given in Appendix II.

5. DYNAMIC ORGANIZATIONAL RATIONALITY: A FRAMEWORK FOR THINKING ABOUT AND MEASURING GROUP EFFECTIVENESS

Our work on the DTDM project began with an intuition that new insights into distributed decision making could be built by exploring analogies between work in social organizations and the operations of large-scale, distributed computer systems. In particular, the Schemer architecture, developed in large part by Dr. Fehling in applications ranging from avionics control to chemical manufacturing, seemed to provide a promising framework under which the various research being undertaken in the DTDM program could be unified.

To begin, we familiarized ourselves with the existing literature in Organization Theory and Organizational Behavior. Pfeffer's extensive review in the 1985 Handbook of the Social Sciences was particularly helpful, as well as various classical works by Weber, March, and Simon. Writings in game theory by Von Neumann & Morganstern, Ordeshook, and Arrow and in team theory by Kim & Roush presented a more technical perspective on group decision problems. Also on the technical side, we gave special attention to mathematical work in organization theory by R.F. Drenick and Alexander Levis in order to understand more thoroughly the dominant theme of organizational modeling work within the DTDM program. Our analysis and insights into the contributions and limitations of this research track were summarized in our Fall 1988 progress report.

Reviewing, discussing and digesting major themes in the literature clarified the key unresolved issues preventing a unification of the many disparate points of view expressed. Missing in particular was any formal notion of the link between organizations and their members or a coherent, general articulation for assessing organizational performance. Keeping these aspects of the organizational problem in mind, we worked for several months conceptualizing, refining and linking together a few primitive notions of organizational rationality. The beginnings of our theory in the deep and very foundational area are presented in the draft paper "Dynamic Organizational Rationality" (Fehling & Lippitz, 1989), which is included as Appendix III to this report.

On the technical side, we applied insights from our foundational work to refine the design of "Schemer," an architecture for distributed- system modeling and analysis, keeping in mind the particular issues of concern in command and control. By examining Schemer's suitability for modeling specific command and control scenarios, we were able to recognize, design, and implement improved problem- solving processes. (This work used the results of basic Schemer design and implementation carried out under IR&D Project 837 at the Rockwell International Science Center).

Based on the theoretical work in "Dynamic Organizational Rationality", we (Fehling and Lippitz) are organizing together with Professor L. Gasser of the Univeristy of Southern California, a Symposium on Organization Theory and AI to be held as part of the 1990

Conference (formerly workshop) on Distributed Artificial Intelligence. Our DTDM work has thus paved the way for what we anticipate will be a very fruitful interaction among fields of study that we believe have not previously been formally linked.

5.1. Research Methodology

It has been said that understanding organizations will be as critical to modern people as understanding agriculture was to our predecessors [Richard Scott, Director, Stanford Center for Organizational Research, in an interview published in The Stanford Daily, 11/14/88]. Yet a clear, unified notion of group problem solving has been elusive for some time, and Organization Theory as a formal discipline is still widely considered to be in a pre-paradigmatic state [Pfeffer]. Therefore, in order to provide a conceptual ground from which specific measures of effectiveness may be derived, we began our inquiry by establishing a departure point based on our most thoughtful consideration of a few foundational questions.

"Why, and under what conditions, do organizations exist?" was the seminal question for our research. Before approaching the "why" part of the question, it was necessary to be clear about what constitutes the "existence" of an organization. In other words, what is it about a group that leads one to recognize it as a unified entity that is different than the simple sum of its parts? Going one step deeper, one must ask, what does it mean to be "unified"? We answer this question with a primitive commitment: For an entity to be considered unified, it must behave in a way that is consistent with its beliefs, values and goals. Beliefs, values and goals - also primitive notions - are derived, through a process of reflection, from examining the coupling between the entity and its environment. The concept of reflective understanding is thus central to our development.

For an individual problem solver, the axioms of rationality provide a basis for judging how well beliefs, values and goals are reflected in behavior. We thus proposed a second foundational question: "What form of intelligence or rationality, if any, can one attribute to organizations?" This question led us to examine the basic problems organizations must overcome to survive and provided a basis for measuring group effectiveness analogous to that which would be applied to individuals.

A deeper consideration of the concepts of goals and values led us to seek out the basic motivations underlying group formation and the criteria which propel its intentional activity. This investigation led naturally to a third foundational question: "How organizations as entities are related to the individuals that constitute them?" Our answer to this question represents the heart of the Organization Metaphor. We express our thesis in terms of a primitive structural notions of "agency" and "autonomy". We also investigated the philosophical implications of the Organization Metaphor and compared it to other theories of group behavior.

We then turned our attention to the functional implications of The Organization Metaphor. Guided by a fourth foundational question - "How do organizations act rationally?" - we investigated the mechanisms agents use to solve their organizational problems. We introduced a primitive notion called "agreements" and develop a theory of their operation.

Taken together, the Organization Metaphor and the Theory of Agreements represent our conception of Dynamic Organizational Rationality. This represents the main contribu-

tion of our work: The concepts of agreements, reflective understanding, rationality, agency, and autonomy provide a framework for studying organizational adaptation, learning and evolution; further consideration of resource constraints provides a formal restatement of sociological theories of power. We used an illustrative example to help build intuition as to how this theory provides a new way of thinking about organizational design and discussed some technical issues concerning the implementation of the theory in computational models of intelligent activity in distributed systems. More details can be found in Appendix III.

6. CONCLUDING REMARKS

Modeling an integrated distributed decision making system in terms of its constituent elements such as communications, computing, data fusion, sensor control, and decision making models is a very complex task. In the two years of this contract period, we identified some important issues in these component areas and investigated them with the long term goal of linking them to each other. We believe that the above results are very useful in developing a comprehensive model for the integrated DTDM environment. We plan to incorporate these results in the future in Schemer, a computational model mentioned above that is being developed under our IR&D, to investigate the possible tradeoffs in an integrated environment.

Much work needs to be done both in the individual areas and in the development of an integrated 'shell.' In the value-based protocol area, we need to investigate such issues as value-based routing, delay estimations, assignment of value-functions in relation to mission objectives, joint value-functions for correlated messages, and implementation issues and costs. In the data fusion and sensor control area, further investigations should consider inclusion of noise in the dynamical model, treatment of unknown time delays between object and sensors, extension to many-object case, and modeling redundancy in the storage of fragments of the state of knowledge. In the cooperative decision making area, apart from expanding on the theoretical notions on Dynamic Organizational Rationality, we need to carefully elucidate the formal basis of Schemer and the mechanism of agreements and develop examples for a number of C^2 systems on that basis. Perhaps the most challenging task would be to formulate an integrated model for C^2 structures and processes using the concepts and results described. This model we seek would provide a common framework within which communications, command hierarchy, data fusion, multi-sensor control, and decision making processes can be described in terms of their real-time interactions in a C^2 environment.

ONR's DTDM program (in which we participated in the last two years) has come to a close (hopefully only temporarily). We look forward to having an opportunity to continue this work if ONR resumes the DTDM program.

APPENDIX I

APPENDIX I

Value-Based Multiplexing of Time-Critical Messages *

L.P. Clare, J.E. Baker and A.R.K. Sastry

Communications and Signal Processing
Rockwell International Science Center
1049 Camino Dos Rios
Thousand Oaks, Ca 91360

Abstract

The concept of message "value" is used as a means for measuring network performance. A "value function" is defined for each message that represents its value to the intended recipient as a function of the total delay in transporting it. These value functions not only incorporate the usual notions of relative priority (i.e., importance) between different messages, but also capture the time-critical characteristics that arise in many applications. Value-based network operation could permeate all layers of communications operation, provided messages are tagged with their value functions. We concentrate in this paper on a single node that multiplexes messages for a single transmitter, and define a value-based multiplexing protocol called Maximum Utility Scheduling for Transmission (MUST). The performance of the MUST protocol is studied through the use of a Markov chain model and through simulations. Results are presented that indicate that MUST's performance is superior to conventional protocols, particularly when traffic arrival statistics are varying or unknown, and loads are high. Both nonpreemptive and preemptive repeat disciplines are considered.

1. Introduction

The level of service provided to a communication network user is typically described in terms of the number of messages that are delivered correctly, and the delays involved. Transcending this is the actual utility or value of the message to the recipient and/or sender. Normally, some messages are more valuable than others, and message values may decrease as their transport delays increase. Prioritized message-handling schemes have been proposed that indirectly account for these factors by treating different message classes differently. However, these schemes

*This work was supported by Office of Naval Research contract N00014-87-C-0703

typically require *a priori* knowledge of the traffic statistics and delay requirements and are not robust to changes in either.

This paper considers a generalization of the usual performance metrics of delay and throughput by specifying "value functions" that represent the utility (or value) of a message to the recipient as a function of the total delay in its transport. We assume that values can be calculated for delivered messages, that values for different users can be compared, and that the objective is to maximize the *total* value delivered. We consider the operation of a network in which each message is tagged with its value function and these tags are used to dynamically allocate the communications resources in order to maximize utility. For simplicity, we concentrate on the value-based operation of a single statistical multiplexer. Messages whose values would be zero can be discarded; in this sense, flow control (congestion control) is being performed. Note that such discards improve performance from the perspective of the network provider, since communications resources are not expended needlessly.

We present in this paper (1) a formal definition of message value by which performance is measured, (2) the synthesis of a realizable value-based protocol called Maximum Utility Scheduling for Transmission (MUST), and (3) numerical examples that indicate that, in certain cases, substantial gains can be achieved by the use of value-based operation in comparison to conventional protocols. In particular, the MUST protocol is very effective when traffic statistics are varying or unknown, when loads are high, or when the system is transmitting a mixture of low priority messages with short deadlines and high priority messages with long deadlines. We feel that the requirements of military and factory communication networks are likely to correspond exactly to the situations where value-based operation promises the greatest advantages.

Our goal in this early effort is primarily to determine whether value-based statistical multiplexing can offer substantial performance enhancement. The conclusion to be drawn is that value-based operation offers performance at least as good as any conventional protocol under any scenario, and can yield substantially improved performance than any conventional protocol in some scenarios. There are many directions for extending the research in value-based network operation, and our work indicates that pursuance is warranted.

Some of the results in this paper were previously presented in [1, 2].

1.1 Message Value

Weaver [3] defined three levels of problems in communications: how accurately symbols can be transmitted (the technical problem), how precisely the symbols convey the desired meaning (the semantic problem), and how effectively the received meaning affects conduct in the desired way (the effectiveness problem). Shannon, in developing information theory, concentrated on only the first of these problem areas, stating [3] that "Frequently the messages have *meaning;* that is they refer

to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem." There have been attempts to extend the theory to include a "qualitative" parameterization of message utility [4, 5], i.e., to account for the third level (effectiveness.) Definitions of "useful self-information" and "useful entropy" were given and some coding theorems were developed for sources having a utility distribution. However, the conclusion [5] was that "the utility concept, as we have formalized it, does not seem to enter deeply in the coding procedures." Here, we are interested in how the utility concept can affect communication operations which operate at a higher level than coding, such as scheduling.

We consider a communications network that is supporting a single overall mission, such as a tactical military operation or factory control. Such a situation provides a simpler setting in which to conceptualize "value," since value is easily associated with the effect that a message reception has in achieving the overall mission objective. Another reason to consider the use of value-based operations in military networks is due to their unique requirements [6] of providing service during crisis situations, when traffic loads and stresses are maximum.

It is clear that the same concepts can be applied to a public switched network that supports many essentially independent communicating user pairs; Pennell [7] has assumed this environment. In this case, a message's "value" is typically perceived differently by different users, perhaps even by the sender and the recipient of the same message. Also, one must distinguish value as perceived by a user from value as perceived by the common carrier. Message value in this setting can be associated with the price that a sender is willing to pay for the service. We will not pursue such economic issues. We assume a cooperative set of users with a common goal.

In this paper, we do not address the construction of value functions themselves, but rather take them as given.

1.2 Message Scheduling

Since message utilities, or values, can vary as a function of their delays, the total value gained by transmitting a collection of messages will depend upon the order in which they are transmitted. This takes us into the general field of scheduling theory, for which there is an extensive literature (e.g., [8, 9, 10, 11]). Forming schedules is known to be typically NP-complete, i.e., computationally difficult. For this reason, a great deal of research has been directed at developing simple scheduling algorithms that are optimal or nearly optimal under specific constraints, such as equally important jobs having various deadlines, or minimizing a weighted sum of completion times, etc. In this paper, we consider the computational requirements to be a secondary issue (although we do present a finite horizon version of the MUST protocol that requires much less processing), and do not constrain the form of the value functions. The rationale is that for many military and factory networks, the

communications resources and end users are more precious than the supporting local computers used to calculate schedules.

The perspective in this paper differs from work by authors in the area of real-time operating systems, where the resource management applies to computing resources. Since the schedules must be computed using the same resource that is being allocated, identification of good, simple heuristic scheduling algorithms is of primary concern. The works of Jensen et al. [12], Locke [13], and Tokuda et al. [14] all consider the process scheduling problem for a uniprocessor [14] or multiprocessor [12, 13] real-time operating system, where each process (task) has a general value function. These papers present heuristic schedulers that are relatively simple to implement, and derive performance results for them as well as a number of conventional scheduling algorithms. Another paper on task scheduling that makes use of a value-based metric is that of Biyabani et al. [15]; their value functions are step functions (hard deadlines) that can be scaled to reflect relative task importance. Message scheduling differs from process scheduling in several respects: preemption is typically costly; message lengths are known; and the cost or value functions can be quite different.

Yemini, in addressing issues in distributed sensor networks, recognized [16] that data "loses its value as time progresses" and that "a real-time protocol may discard outdated messages and/or give priority service to messages according to their 'value.'" A number of other authors have investigated systems of this type, including multiplexing of voice with congestion-dependent fidelity [17, 18] and multiple access protocols for time constrained messages [19, 20]. Pennell [7] considered a multiplexer with a finite capacity buffer that handles messages with linearly decreasing value, and derived algorithms for scheduling service and for choosing which message to discard when buffer overflow occurred. Pennell also investigated value-based routing. Kurose and Towsley [21] recently analyzed scheduling policies for multiplexing real-time traffic (with hard deadlines) and non-real-time traffic. The objective was to maximize the number of real-time messages that meet their deadlines as well as minimize the mean delay for the non-real-time messages. Panwar et al. [22] and Bhattacharya and Ephremides [23] investigated the optimality of the shortest time to extinction (STE) rule in dynamic scheduling of messages with deadlines (and otherwise equal values).

In the next section, we define the system model and describe the operation of the MUST protocol with either infinite and finite horizons. Our performance evaluation approach is given in Section 3. Numerical results from simulations are given in Section 4. Both nonpreemptive and preemptive repeat disciplines were considered as well as modifications of the conventional protocols in which valueless messages are discarded without being transmitted. Section 5 presents an analytical model of a slotted nonpreemptive system operating under the MUST protocol, and the compares the system performance to that under other protocols. The results in sections 4 and 5 show that MUST's performance is superior to any of the conventional protocols of FIFO, LIFO, or fixed priorities. Conclusions and possible

extensions of the work are given in Section 6.

2. System Model and MUST Protocol

The basic concept of handling messages in a communication system based upon their utility is illustrated in the context of statistical multiplexing, which consists of the following elements. There is a single transmitter (single channel) with a single infinite capacity buffer (when we later consider a Markov chain model of the system, we will assume that the buffer has limited capacity.) Messages arrive at random times, causing a queue to occasionally form. The message lengths can be variable, but are assumed to be known at the time of arrival, so that it is known how long it will take to transmit each message. Each message is tagged with a deterministic *value function* that indicates the value (or utility) that the message has to the receiver as a function of the time between the message arrival at the transmitter and the completion of the message transmission. All messages are assumed to be autonomous in the sense that their value to the receiver depends only on the delay in transport of that message, and does not depend on the delivery characteristics of other messages.

The queueing discipline chooses the next message in the buffer that is to be transmitted. Messages may also be removed from the buffer and discarded. In addition, a message being transmitted can be preempted, and either placed back in the buffer or discarded. A preempted message must be completely retransmitted, i.e., preemptive repeat is assumed. We seek to synthesize a scheduling policy, and possible preemption procedure, so that the rate of received message utility is maximum.

We define the following notation. Let

$X(t)$ = number of messages at the transmitter (in the buffer or in transmission).

A_x = arrival time of the x^{th} message at the transmitter, $x = 1, 2, \ldots, X$. The indexing of the messages is such that $A_1 < A_2 < \cdots < A_X$. The vector $\underline{A} = (A_1, A_2, \ldots, A_X)$ changes with time t as new messages arrive and old messages are removed because of being transmitted or discarded. The messages in the system at time t are identified by the indexing based upon order of arrival.

x^* = index of the message currently being transmitted. If no transmission is ongoing, take $x^* = 0$. For example, the FIFO discipline would have $x^* = 1$, and the LIFO discipline with preemption would have $x^* = X$.

$V_x(\cdot)$ = value function for the x^{th} message at the transmitter. (Again, the index depends on the time t.) $V_x(D)$ is the value for the x^{th} message if its delay is D, where delay is defined as the time since the message arrived at the transmitter until it completes transmission. The value functions are assumed to be monotone nonincreasing and nonnegative. Every message could have a distinct value function. Note that if all value functions are scaled by the same positive number, then there should be no difference in the sequence generated by

the value-based queueing policy, and the total value received is simply scaled accordingly. In this sense the values are only relative.

L_x = length of time to transmit the x^{th} message (i.e., the service time for the message, and does not include the queueing time).

$R(t)$ = duration of time since the ongoing transmission began. If there is no transmission active at time t, then take $R(t) = 0$. The quantities x^* and R are needed in the discussion because of the possibility of preemption of transmissions.

2.1 The MUST Protocol

We propose the following procedure, referred to as the Maximum Utility Scheduling for Transmission (MUST) protocol.

Messages arrive at a multiplexer and wait for transmission. Their arrival times and lengths are governed by arbitrary and possibly unknown stochastic processes. Each arriving message has an associated value function, that gives the value of the message to system as a function of its delay (time in system.) Whenever a new message arrives, or a transmission is completed, the scheduler considers the entire collection of waiting messages (plus the message in service if preemption is allowed) and considers each possible transmission sequence (i.e., each possible permutation of the waiting messages) *assuming that no further messages arrive in the system.* Since message lengths are known, the scheduler can determine when the transmission of each message in a sequence will occur, and, thus, the delay and corresponding value for each message. The scheduler determines the value-maximizing schedule and transmits according to it. If a new message arrives, the tentative schedule is recomputed. The scheduler knows when messages reach a value of zero and discards them. If a message is preempted, the entire message will have to be repeated.

If the multiplexer knew when all future arrivals would occur, it could, in principle, determine a transmission schedule that maximized the value delivered (for the given sample path.) However, in most practical systems this is impossible. At best, a partial statistical description of the message arrival processes may be available. A number of authors have considered specific sets of priorities and assumptions about the statistical nature of the arrival processes (e.g., Poisson arrival streams with exponentially distributed message lengths and hard deadlines) and determined optimal scheduling rules [24, 25, 23, 26]. However, we are particularly concerned with the performance of the scheduler under surges of traffic, overload conditions, and changes in traffic patterns. Therefore, the protocol makes no assumptions about future arrivals, and considers only the messages already in the system.

The MUST protocol is not necessarily optimal for any particular arrival process, but can generally be expected to yield excellent performance compared to other nonanticipatory policies (i.e., policies that do not incorporate probabilistic or sample path knowledge of the arrival process.) It is realizable, although it can require considerable computation if the buffer size X is large (about ten or more).

We define the following notation.

$\underline{\pi} = (\pi_1, \pi_2, \ldots, \pi_X)$ = a permutation of the integers $\{1, 2, \ldots, X\}$. This defines a tentative schedule of the X messages at the transmitter, where the messages are indexed in the order of their arrival, 1 being the oldest, so that the next message to be transmitted is the one with index π_1. If no further arrivals occur, the subsequent messages would be transmitted in the order π_2, π_3, etc. The tentative schedule is recomputed each time an arrival occurs, so that $X = X(\text{arrival time}^+)$, and each tentative schedule $\underline{\pi}$ remains in effect until the next arrival occurs.

In a selected tentative schedule $\underline{\pi}$, it is quite possible that the last few messages ($\pi_k, \pi_{k+1}, \ldots, \pi_X$, for some k) would have zero value by the time they were transmitted in that sequence. This might seem to indicate that these messages should be discarded immediately. However, $\underline{\pi}$ is a *tentative* schedule, and it is possible that a later arrival will cause a new schedule to be formed in which one or more of the messages that would have been discarded now will be transmitted with positive value. We therefore do not discard messages based upon tentative schedules. However, the queueing discipline is assumed to discard messages as soon as there is no hope of them being transmitted with positive value, i.e., the x^{th} message is discarded at the first time t for which $V_x(t - A_x + L_x) = 0$. In an actual implementation, the discipline could wait for certain times before throwing out useless messages, such as cleaning up the buffer every arrival and departure instant.

At the instant t of a new arrival, the new tentative schedule is computed as follows. Assuming that X includes the new arrival ($X = X(t^+)$), there are $(X-1)!$ possible schedules in which the message currently being transmitted will continue to be transmitted without preemption, and another $X! - (X-1)!$ permutations that preempt the message being transmitted in favor of another. For each permutation that preempts the message being transmitted, the total value under the assumption of no further arrivals is

$$V_{\text{Total}} = \sum_{x=1}^{X} V_{\pi_x}\Big([t - A_{\pi_x}] + \sum_{n=1}^{x} L_{\pi_n}\Big).$$

For each permutation in which the message in transmission is not preempted, so that $\pi_1 = x^*$, the total value would be

$$V_{\text{Total}} = \sum_{x=1}^{X} V_{\pi_x}\Big([t - R(t) - A_{\pi_x}] + \sum_{n=1}^{x} L_{\pi_n}\Big).$$

It is possible that multiple permutations $\underline{\pi}$ will yield identical maximum total values. We then add some degree of anticipation to the discipline by choosing that schedule that reaches the maximum total value first. If a tie still exists, and one or more of these total value maximizing permutations has the ongoing transmission continue ($\pi_1 = x^*$), then eliminate all other permutations. Further ties are broken using a heuristic that is related to the form of the value functions, such as shortest deadline first.

2.2 MUST with Finite Horizon

When the queue size is large, the computational burden required by MUST is overwhelming, even with modern computer technology. Intuitively, a large queue size implies that a heavy traffic rate is present, so that a new arrival is likely to occur well before all of the messages presently in the tentative schedule are transmitted. We conjecture that we should obtain approximately the same performance if, rather than forming complete schedules for all of the X messages in the queue, we instead only form partial schedules for the next n messages to be transmitted, where n is much smaller than X but large enough to expect that another arrival will occur before all n are transmitted. That is, we consider each possible selection of n out of X messages, and each possible permutation of each set of n. The computational burden is reduced from evaluating $X!$ schedules to $\frac{X!}{(X-n)!}$.

This is essentially the idea behind MUST with a finite horizon. The only slight complication is that in order to compare schedules fairly, one must compare them against a common completion time, and not against the same number of messages (obviously these are equivalent if all messages are the same length). We define the following notation:

h = horizon [time] for the MUST protocol.

i^* = number of messages within the horizon time; this depends on the permutation selected. i^* satisfies

$$\left(\sum_{i=1}^{i^*} L_{\pi_i}\right) - R(t) \leq h, \quad \text{and} \quad \left(\sum_{i=1}^{i^*+1} L_{\pi_i}\right) - R(t) > h.$$

The operation of MUST with finite horizon h requires that a new tentative schedule is recomputed each time a new arrival occurs *and* after each departure (whenever $i^* < X$ for any permutation $\underline{\pi}$) using

$$V_{\text{Total}} = \sum_{i=1}^{i^*} V_{\pi_i}\left([t - R(t) - A_{\pi_i}] + \sum_{n=1}^{i} L_{\pi_n}\right).$$

Of course, if $h = \infty$ then we always have $i^* = X$ and the previously defined MUST protocol results. Figure 1 depicts how two competing schedules are compared in terms of the total value delivered.

We have investigated through simulation many different traffic scenarios and generally found that when h is just large enough to correspond to about three message transmission times, we obtained performance essentially equivalent to that of the unconstrained MUST protocol ($h = \infty$). This represents a tremendous savings in computation. For example, if $X = 10$ and h is such that the typical schedule length is $n = 3$ then $X! = 3{,}628{,}800$ while $\frac{X!}{(X-n)!} = 720$.

2.3 Comments

It is possible to find cases for which the MUST protocol will yield a lower total value than some other scheduling protocol. The suboptimal nature of the MUST protocol can arise from its inability to see into the future, and act accordingly. It may be possible to improve the protocol by attempting to estimate future events based upon some a priori knowledge on the arrival statistics and the corresponding value functions and message lengths. For example, two tentative schedules may yield slightly different total values, but the one with the lower total value may complete much sooner; this then frees up the transmitter earlier for use by potential future arrivals, and therefore may ultimately be the better choice (i.e., yield larger value in the long term.)

Intuitively, a case where MUST will be superior to all conventional protocols is when we have mixtures of hard deadline value functions where some are of high value with long deadlines and others are of lower value but with shorter deadlines. We feel that this may be representative of a military requirement, as is illustrated in Figure 2. This Figure suggests that messages generated at higher echelons will be more important, but will generally involve wide area operations of relatively long time spans and therefore be tolerant of relatively longer delays, while lower echelon traffic will be less important but will require quick response times. It appears that no simple rules exist that provide optimal performance for this type of traffic; only a protocol like MUST can adequately handle these requirements.

3. Performance Evaluation Approach

The performance of a specific queueing discipline can be measured by computing the expected total value delivered over a representative time interval, where the expectation is taken with respect to the arrival process. In the event that the arrival process is time-stationary, the time-averaged rate of received value can exist and be used as a performance measure. Note that if the arrival process is stationary and all value functions are zero beyond some delay (i.e., $\exists\ t_{\text{deadline}} < \infty$ such that $V(t^{+}_{\text{deadline}}) = 0$), then the buffer size process is ergodic when the MUST protocol is used.

We use two complementary performance evaluation approaches: simulation and analytical modeling. The simulation model allows general value functions, continuous time operation, and preemptive or nonpreemptive service. For reasons of tractability, the analytical model is limited to a slotted system with a fixed number of message classes and nonpreemptive service.

4. The Simulation Model

The simulation of the baseline protocol has the same difficulties as would an actual implementation in terms of the computational burden of comparing $X!$ schedules at each arrival. Our simulation of the baseline protocol currently employs a brute force procedure that exhaustively enumerates all possible cases, and chooses the best one. For a slotted system with fixed message lengths and a finite buffer size, only a finite number of possible system states are possible, and schedules can be calculated in advance and stored in lookup tables. This approach is used in the analytical model described below.

To obtain a quantitative feeling for the relative performance of the MUST scheduling policy, we have conducted extensive computer simulations and compared it with the conventional policies of First-In-First-Out (FIFO), Last-In-First-Out (LIFO), and Head-of-Line (HOL), also known as strict or fixed priority. Furthermore, we have considered two variations that can occur: whether or not preemption of the ongoing transmission is allowed, and also whether the protocol discards (without transmission) all messages that would have zero value by the time its transmission completed. The specific protocols simulated were: (1) FIFO, (2) FIFO with discarding of valueless messages, (3) nonpreemptive LIFO, (4) nonpreemptive LIFO with discarding, (5) nonpreemptive HOL, (6) nonpreemptive HOL with discarding, (7) preemptive repeat HOL, (8) preemptive repeat HOL with discarding, (9) nonpreemptive MUST with possibly finite horizon h, (10) preemptive repeat MUST with possibly finite horizon, and (11) preemptive Shortest Time to Extinction.

The performance will depend on the arrival process. Clearly, the largest differences between MUST and any other protocol will occur when queue sizes at least occasionally become moderately large, since otherwise there is essentially no choice to be made in scheduling. Large queues result if either the arrivals occur in bursts or the load (arrival rate to service rate ratio) is high. We could contrive a burst arrival process that makes MUST look arbitrarily better than other disciplines, but we instead chose the more common assumption of Poisson arrival processes. We generally selected the aggregate load to be 100%, so that sizable queues will occur. The simulation runs were made for a simulated time interval of one second, with a channel transmission rate of 1Mb/s and packet lengths nominally of 1000 bits. For all but the FIFO case without discarding, the results are representative of steady state conditions.

The MUST protocol allows complete freedom in selecting value functions associated with the messages. However, in the numerical examples shown in this paper, we limit ourselves to only two simple forms for the value functions: hard deadline and linear decay. Each such function is defined by two parameters, the initial value and the deadline.

4.1 Simulation Results

All of the results presented are limited to only two types of traffic sources, where all messages generated by a given "source" have the same value functions. Figure 3 illustrates the performances of the various protocols when the two types of value functions are linear as shown and all messages are the same length of 1000 bits (this is taken to be unit length). Performance is compared as a function of the deadline for the messages. The MUST protocol with preemption is consistently the best, although for this case the HOL discipline with preemption and discarding of valueless messages is almost as good. In fact, we generally found that discarding of valueless messages for any protocol results in considerable performance gains. (Of course, this requires the messages to be tagged with their deadlines.)

Figure 4 compares the various protocols when the value functions for all messages are the same as shown but the two sources differ in terms of the message lengths. We take the ratio of the message lengths to be unity, and plot the performance as the ratio of message lengths is varied. Again, MUST performs at least as well as all the other protocols over the entire range. LIFO with discarding performs well when there is little message length disparity and HOL with discarding performs well as the disparity increases. If the message length disparity is time-varying then MUST's performance will be superior to the other protocols. In general, MUST is robust in that the traffic characteristics need not be known before hand and may vary over a wide range.

Figure 5 illustrates the performance of MUST when the value functions are equal value hard deadlines with different deadlines.

The performance of MUST as a function of the horizon h is illustrated in Figure 6. Two cases are considered, with two traffic source types each. The value function forms are shown. In both cases it is clear that a horizon value of only about 2 message transmission times is adequate to approach optimal performance. When very small horizon times are chosen, many ties occur among the schedules, and the heuristic for breaking ties becomes important. In these examples we simply used a FIFO rule for breaking ties, so that the performance as $h \rightarrow 0$ is equivalent to FIFO with discarding.

5. The Analytical Model

The system model used in the simulations of the MUST protocol is very general and not amenable to an exact analytical evaluation. Therefore, we will present a less general model that can be modeled analytically.

5.1 Model Features

In particular, we consider a discrete time model of a multiplexer. The time axis is *slotted*. All message transmissions begin at the start of a slot and continue for an

integral number of slots. Message lengths and delays are measured in slots.

The multiplexer is modeled as a single server queue. Messages arrive at the queue according to a random process. Each message has an associated value function. We restrict messages to belong to one of several classes. All messages in a class have the same length and the same value function.

The server transmits messages nonpreemptively. At the end of a message service, the server chooses another message for service according to the scheduling protocol; or if there are no ready messages, the server remains idle until a slot contains arrivals. A message is then chosen for service from among those arrivals according to the scheduling protocol. The scheduling protocol is implemented as a probability of selection for service for each waiting message, for each possible buffer state. This allows us to easily compare a wide range of scheduling rules (including MUST.)

If a message's value function reaches zero while it is still waiting for transmission, the message is discarded.

The queue size is limited for each message class. At most N_c class c messages can be waiting for service at one time. If messages need to be discarded to stay under the limit, the oldest messages are discarded and new arrivals are kept.

5.1.1 The Environment

We consider the operation of a multiplexer in a changing environment, where traffic conditions may vary rapidly, traffic sources may be inter-related, and overloads may occur. Message value functions could also depend upon the environment, but this is not explored in the current paper.

The random environment is modeled by a finite state, irreducible and aperiodic Markov chain (Figure 7 shows a specific example of a random environment) that moves through a state space $E = \{e_1, e_2, \ldots, e_K\}$ according to a state transition matrix P_E. The environment can change from slot to slot. The initial state distribution for the environment is given. By using a Markovian model of the environment, we can study the performance of the multiplexer under highly varying traffic loads and varying conditions, while retaining analytic tractability.

5.1.2 The Arrival Processes

Each message that arrives at the multiplexer belongs to one of C classes. Messages that arrive during a slot are recorded at the end of the slot. Messages of class c all have the fixed length (i.e., required transmission time) of b_c slots. Variable message lengths could be considered by breaking a class into separate classes, each with fixed length messages. Note that the scheduler is assumed to know the lengths of the messages waiting in its queue. Many scheduling papers have assumed that message lengths are exponentially distributed, and unknown by the scheduler, which is an unrealistic assumption for many systems.

The joint distribution of the number of messages of each class that arrive in a single slot can depend upon the value of the environment variable for that slot. Conditioned upon the value of the environment, the joint distribution is independent of the past and future of the system. Thus, we have a Markov-modulated joint arrival process (see, e.g., [27].) Message arrivals from different classes are allowed to be correlated within a slot, and, through the state of the environment, from slot to slot.

Let the single slot arrival probabilities, conditioned upon the state of the environment be:

$$a(j_1, j_2, \ldots, j_C|e) = P\{\text{no. of arrivals of class } k = j_k, k = 1, 2, \ldots, C \\ |\text{the state of the environment is } e\}$$

where $0 \leq j_k \leq N_c, k = 1, 2, \ldots, C$.

Of course,

$$\sum_{j_1, j_2, \ldots, j_C} a(j_1, j_2, \ldots, j_C|e) = 1 \qquad \forall e \in E$$

5.1.3 Message Ages and Phases

Message values generally decrease as they age. Voice packets or control messages may have hard deadlines, a message containing position information about a moving object will become less accurate as time passes, and so on. Therefore, message phases (a generalization of message age) are tracked for use by the MUST protocol.

For each message class c, messages belong to one of M_c *phases.* A newly arriving message belongs to phase 1 and the only allowed phase transitions are $1 \Rightarrow 2 \Rightarrow \ldots M_c \Rightarrow$ discard. (An exception to this occurs when messages are discarded to stay under the limit of N_c messages in class c.) Message transitions *from* phase M_c correspond to the discard of messages that have become worthless. See Figure 8.

Class c messages in the same phase are interchangeable, in the sense that their values are identical.

Phase transitions occur randomly, and the probabilities may depend on the state of the environment. Conditioned upon the state of the environment, message transitions occur independently. Let $p_c(j|e), j \geq 1$, be the probability that a class c message that is in phase j at the start of a slot will transition to state $j + 1$, given that the environment state for the slot is e. With probability $1 - p_c(j|e)$, the message will remain in phase j.

Note that:

- If all the probabilities $p_c(j|e)$ for a particular class equal 1, then the phase will just be the message age.
- We can trade off accuracy against computational complexity by approximating a larger number of ages by a smaller number of phases. This is, of course, the primary reason for introducing the concept of phases.

- If, for some c and j, $p_c(j|e) = 0 \quad \forall e \in E$, then class c messages that reach phase j will remain there. Thus, in particular, a message class whose value function does not depend on age can be modeled with a single phase that the messages never transition out of.

- If the transition probabilities do not depend on the state of the environment, messages will spend a geometrically distributed number of slots in each phase.

Figure 9 shows the interaction between the environment and the message arrival and transition processes.

5.1.4 Value Functions

The value function is a deterministic function of message phase at delivery and is the same for all messages of a given class. (The value could also depend on the state of the environment at delivery, but that is not considered in this paper.) The value function gives the intrinsic value of a message to the system as a function of time (or phase.) The objective of MUST is to maximize the total value delivered by the system. For each class, the value of a message is a nonnegative, nonincreasing function of the message's phase.

Let $V_c(j)$ be the value of a class c message delivered to its destination while in phase j.

The above assumptions can be written as:

$$V_c(j) \geq 0 \qquad c = 1, \ldots, C;\ j = 1, \ldots, M_c$$
$$V_c(j) \geq V_c(j+1) \qquad c = 1, \ldots, C;\ j = 1, \ldots, M_c$$

5.1.5 Message Selection – The MUST Protocol

At the end of each message transmission, or at the end of an idle slot, the MUST protocol either selects a message for service or keeps the system idle for one slot. (If, e.g., a single message is waiting for service, but its transmission will take two slots while its value will reach zero at the end of the first slot, the system will remain idle.)

Suppose that some number of messages are waiting in the queue. The MUST protocol considers each distinguishable ordering of the messages (orderings in which messages of the same class and phase are interchanged are not distinguishable) and calculates the expected value that would be delivered by transmitting the messages in that order. Because of possible transitions in the environment variable and phase transitions by the messages, the value delivered by a particular ordering is not deterministic. Rather than tracking or attempting to predict the environment variable, the conditional phase transition probabilities are weighted by the steady state distribution of the environment variable and are combined. The combined

phase transition probabilities are then used in the calculation of the expected value of a particular schedule. Note that if the phase transition probabilities are the same for all values of the environment variable, the weighting is unnecessary. If, additionally, the phase transition probabilities are all 1 (i.e., phase is just age) the value calculated for each sequence is deterministic.

Note that the protocol does not consider future arrivals. The benefit of this is that no assumptions need be made about the arrival processes. MUST is thus very suitable for conditions where the traffic conditions are varying or unknown. However, a value-based protocol incorporating knowledge about traffic conditions could be designed.

Note that for a slotted system with a limited number of classes and a limited buffer size, the message selection function can be computed ahead of time and stored as a lookup table.

5.1.6 The State

The state of the system consists of the environment state and the number of messages of each class and phase value that are waiting for transmission. We assume that there is an upper bound to the number of messages of each class allowed in the system at one time (rather than a bound on the total number of messages.) In particular, at most N_c messages of class c are allowed in the system at one time. If enough new class c messages arrive during a slot so that the total number of class c messages would exceed N_c, messages are discarded, starting with the least valuable (since value is nonincreasing, the oldest are discarded.)

For C message sources, where class c has M_c nonzero value states, there are

$$|E| \cdot \prod_{i=1}^{C} \binom{M_c+N_c}{N_c}$$

possible states.

For the nonpreemptive version of MUST, transmission decisions are made at the end of each message transmission or idle slot. Since there are a finite number of possible system states, each state can be mapped to a corresponding integer. Let X_i be the (integer corresponding to the) state of the system just prior to the completion of the ith message transmission or idle slot.

Upon observing the state of the system, the MUST protocol (deterministically) decides which message to transmit, if any; or if the system is empty, to remain idle for one slot. Let Y_i be the (integer corresponding to the) state of the system just after the selection decision has been made. For the purposes of the model, the decision is assumed to be instantaneous. If the system is empty, X_i and Y_i will be the same. Otherwise, the only difference between X_i and Y_i will be that a single message (the one chosen for transmission by the MUST protocol) will have been removed for transmission. Note that the message is removed from the state at the start of transmission, not at the end.

5.2 Transition Probabilities and Stationary Distributions

Consider a transition from X_i to X_{i+1}. This can be separated into the transition from X_i to Y_i, followed by the transition from Y_i to X_{i+1}.

Let M be the transition matrix between X_i and Y_i, i.e., the matrix showing the change in state due to the MUST decision. Since the decision is a deterministic function of the state, each row of M (corresponding to the value of X_i) will have all zero entries except for a single 1 in the column corresponding to the Y_i value due to the MUST protocol.

Let R be the one slot state transition matrix for the system, excluding any message in service, i.e., it expresses the transition from Y_i to X_{i+1}. The transitions in R consist of four parts:

1. The environment changes state according to its transition matrix;
2. Given the new value of the environment, each queued message independently either stays in its current phase, or makes a transition to the next (or out of the system if it is in the last phase) according to the appropriate probability (a function of the new value of the environment, the message class, and the message phase);
3. New messages arrive at the multiplexer according to the arrival distribution (which depends only upon the environment);
4. If the number of messages in a class exceeds the maximum number allowed, messages are discarded, starting with the lowest value messages (even before their value reaches zero). Note that this is an assumption used to keep the model computationally tractable, not a fundamental feature of the protocol.

Suppose that the transition from X_i to Y_i corresponds to the transmission of a class c message, of length b_c. Then the distribution of the state X_{i+1}, i.e., the state at the end of the message transmission, given Y_i and the class of the message in service, is given by the Y_i-th row of the matrix R^{b_c}. If the channel remains idle, the distribution is given by the X_i-th row of R. See Figure 10 for an example.

By construction, $X = \{X_i, i = 1, 2, \ldots\}$ forms a discrete time Markov chain on a finite state space. Because the state space is finite, the environment process is aperiodic and irreducible, and the arrival processes are conditionally independent from slot to slot, X is aperiodic and a stationary distribution exists. (The stationary distribution may not be unique. By eliminating transient states and considering different sets of communicating states separately, existence of unique stationary distributions for each set of communicating states can be guaranteed.)

Let P_X be the one-step transition matrix for X.

$$P_X = \{P_X(j,k) = \text{Prob}(X_{i+1} = k | X_i = j)\}$$

We can construct P_X by repeating the following process for each possible state, say $X_i = j$:

1. Look up the MUST decision corresponding to $X_i = j$ in the matrix M. Suppose that $Y_i = k$, and the class of the message selected for transmission is c.

2. Set the j-th row of P_X equal to the k-th row of R^{b_c}.

Let ρ_X be the stationary distribution for $\{X_i, i \geq 0\}$. (If there are multiple sets of communicating states, each set can be considered in turn.) Let ρ_{X_0} be the initial distribution for X_0.

Once P_X has been calculated, ρ_X can be found using the relationships:

$$\rho_X = \rho_X P_X, \qquad \rho_X \cdot \mathbf{1} = 1 \tag{1}$$

where $\mathbf{1}$ is a vector of 1's.

5.3 Numerical Solution Techniques

For a given set of parameter values, the equations (1) for ρ_X are solved in the following manner:

1. Transient states are eliminated. Particular choices for the arrival distribution and other parameters can make some states unreachable, so those states are eliminated.

2. The states are reordered so that states that could be reached in a single step are given index values near each other.

3. In the reduced and renumbered version of equation (1), the left and right hand sides of the first equation are combined and the transpose is taken, to give a set of homogeneous equations satisfied by the remaining elements of ρ_X. The restriction that the elements of ρ_X sum to one can be incorporated by eliminating one element of ρ_X by setting it equal to 1 minus the rest (yielding a nonhomogeneous set of equations.) For the iterative methods used below this last step is not necessary since the iterates of ρ_X will maintain a norm of one except for round-off, and can be renormalized whenever necessary.

4. At this point we have a large, sparse set of linear equations. A large literature exists on the solution of such sets of equations [28, 29, 30, 31]. A smaller number of papers cover the specific topic of the numerical solution of Markov Chains [32, 33, 34, 35, 36, 37, 38]. The examples in the performance analysis section were calculated using the Biconjugate Gradient-Squared Method [39] as implemented in the NSPCG numerical package [31].

5.4 Performance Measures

For each state X_i, MUST tells us which message is to be selected for service, and, since message lengths are fixed, the length of time until the next decision point. Knowing the state of the environment and the starting class and phase of the message in service, we can calculate the distribution of the value delivered at the end of the interval. The primary performance measure is mean value delivered per slot. The overall mean, the mean for each class, and the conditional mean for each environment state are also calculated.

5.5 Numerical Examples

In this section, the performance of the MUST protocol under a variety of conditions is investigated and compared to other scheduling disciplines. Specifically, it is compared to

- FIFO: first in, first out;
- LIFO: last in, first out;
- ED: earliest deadline first (message lengths aren't considered in the selection);
- STE: shortest time to extinction;
- HOL: head of line.

For FIFO, LIFO, and HOL, both the case where the disciplines completely ignore the value functions, and the case where they discard messages that have reached a value of 0 are considered. The comparison to these disciplines gives a measure of the benefit achievable through the use of value functions, whether through the MUST protocol or through a simpler value-based heuristic. The various protocols were implemented by calculating the appropriate decision matrix (M) for each protocol.

Case 1 - Hard Deadlines

We first consider a case with two message classes, each with a hard deadline. Class 1 messages have a length of 1 slot and a hard deadline of 3 slots. A class 1 message delivered within its deadline yields a value of 1. Class 2 messages have a length of 2 slots and a hard deadline of 6 slots. A class 2 message delivered within its deadline yields a value of 2 (see Figure 11 for the value functions.)

Each class can have a maximum of 4 messages in the queue at one time. Messages of each class arrive according to independent truncated geometric distributions. The maximum number of new arrivals in a single slot is 2 for each class. We compare the performance of MUST to LIFO and FIFO, both with and without discarding messages that have reached 0 in value, and to HOL, where class 2 is

given priority (which is the best HOL case) both with and without discard. We also consider STE and ED with discard. In Figure 12 we hold the overall arrival rate at $\rho = 1$ while we vary the mix of traffic due to each class. In Figure 13 we hold the arrival rate constant for class 1 while increasing the arrival rate for class 2. In both Figures, we can see that the performance of MUST is superior to the other protocols. In Figure 14 we examine the proportion of value delivered for each class as the class 2 arrival rate changes.

In Figure 15 we compare the values calculated for the MUST protocol using message ages, to those calculated using a reduced phase representation. Specifically, groups of two phases (where messages stay in each phase for exactly one time unit) are combined into a single phase with a geometrically distributed length of stay, with mean two.

The reduced phase representation significantly reduces the state space dimension. Figure 15 shows that results from the reduced representation are close to those of the exact case, particularly at lower arrival rates.

Case 2 - Linearly Decreasing Value Functions

We next consider a case with linearly decreasing value functions. The value functions are shown in Figure 16. There are two message classes, each with message length 1, a maximum of 4 messages allowed in the queue at one time, and truncated geometric arrivals with a maximum of two arrivals per slot. In Figures 17–19 we can see the effects of changing the relative message values.

Case 3 - The Effects of Ignoring Future Arrivals

In this case, an optimal decision based upon messages currently waiting for service is not necessarily globally optimal. That is, for particular (fairly extreme) parameter values and arrival rates, other protocols can perform better than MUST. Note, however, that one would have to know the arrival rates and other parameters in order to pick the appropriate protocol.

We consider a case with two classes of messages, each with a linearly decreasing value function (see Figure 20.) In Figures 21–23 we see that for some choices of parameters, LIFO with the discard of messages that reach zero can give better performance than MUST. This is easily explained. Suppose that the queue held a single class 2 message of age 1, and a single class 1 message of age 3. Under the MUST protocol, the scheduler would transmit the class 1 message, to be followed by the class 2 message (since the total value achieved would be higher than transmitting the class 2 message and losing the class 1 message.) On the other hand, the LIFO scheduler would transmit the class 2 message. Since the arrival rate of class 2 messages is very high, with high probability a new class 2 message will arrive for the next slot. Thus, the best action would have been to transmit the class 2 message.

6. Conclusions and Extensions

We have shown that the concept of assignment of value functions for time-critical messages is a useful one that can be exploited in designing efficient scheduling protocols. We have developed a protocol called Maximum Utility Scheduling for Transmission (MUST) that appears to be very efficient and robust under different types of traffic conditions. The quantitative results described are based on both simulation and analytical models, and have given an excellent preview of the potential performance capabilities of MUST. In our future work, apart from developing these results further at a node level, we will consider their extension to the network level, introducing such issues as value-based routing, delay estimations, assignment of value functions in relation to mission objectives, joint value functions for correlated messages, and implementation issues and costs.

Acknowledgments

The authors wish to thank C.-Y. Wang for the valuable support provided in the coding, validation, and execution of the computer simulation models. We also wish to thank Jonathan Agre and Robert Doyle for many stimulating discussions.

7. References

[1] L. P. Clare and A. R. K. Sastry, "Value-based multiplexing of time-critical traffic," in To appear in *Proceedings of Milcom '89*, 1989.

[2] J. Baker, "Value-based message multiplexing: An analytical model." Submitted.

[3] C. Shannon and W. Weaver, *The Mathematical Theory of Communication.* Urbana, IL: The University of Illinois Press, 1949.

[4] M. Beliş and S. Guiaşu, "A quantitative-qualitative measure of information in cybernetic systems," *IEEE Transactions on Information Theory*, vol. IT-14, pp. 593–594, July 1968.

[5] G. Longo, *Quantitative-Qualitative Measure of Information.* New York: Springer-Verlag, 1972.

[6] V. G. Cerf and R. E. Lyons, "Military requirements for packet switched networks and their implications for protocol standardization," *Computer Networks*, vol. 7, pp. 293–306, 1983.

[7] J. P. Pennell, *Message Value Metrics: A Technique for Optimum Flow Control in Data Communication Networks.* PhD thesis, University of Virginia, January 1988.

[8] K. Baker, *Introduction to Sequencing and Scheduling.* New York: John Wiley and Sons, 1974.

[9] J. E. G. Coffman, *Computer and Job-Shop Scheduling Theory.* New York: John Wiley and Sons, 1976.

[10] A. H. G. R. Kan, *Machine Scheduling Problems.* The Hague: Martinus Nijhoff, 1976.

[11] S. French, *Sequencing and Scheduling.* New York: John Wiley and Sons, 1982.

[12] E. D. Jensen, C. D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems," in *Proceedings of IEEE Real-Time Systems Symposium*, (San Diego, California), December 1985.

[13] C. D. Locke, *Best-Effort Decision Making for Real-Time Scheduling.* PhD thesis, Carnegie Mellon University, May 1986.

[14] H. Tokuda, J. W. Wendorf, and H.-Y. Wang, "Implementation of a time-driven scheduler for real-time operating systems," in *Proceedings of IEEE Real-Time Systems Symposium*, (San Jose, California), December 1987.

[15] S. R. Biyabani and J. A. Stankovic, "The integration of deadline and criticalness in hard real-time scheduling," in *Proceedings of the Real-Time Systems Symposium*, pp. 152–160, December 1988.

[16] Y. Yemini, "Distributed sensors networks (dsn): An attempt to define the issues," in *Proceedings of the Workshop on Distributed Sensor Networks*, (Carnegie-Mellon University), pp. 53–60, December 7–8 1978.

[17] T. Bially, B. Gold, and S. Seneff, "A technique for adaptive voice flow control in integrated packet networks," *IEEE Transactions on Communications*, vol. COM-28, pp. 325–333, March 1980.

[18] B. G. Kim and D. Towsley, "Dynamic flow control protocols for packet-switching multiplexers serving real-time multipacket messages," *IEEE Transactions on Communications*, vol. COM–34, pp. 348–356, April 1986.

[19] J. F. Kurose and M. Schwartz, "A family of window protocols for time constrained applications in csma networks," in *Proceedings of IEEE INFOCOM 83*, (San Diego, California), pp. 405–413, April 18–21 1983.

[20] W. Zhao and K. Ramamritham, "Virtual time csma protocols for hard real-time communication," *IEEE Transactions on Software Engineering*, vol. SE-13, pp. 938–952, August 1987.

[21] J. F. Kurose and D. Towsley, "Scheduling policies for real-time and non-real-time traffic in a statistical multiplexer," in *Proceedings of IEEE INFOCOM'89*, (Ottawa, Ontario), pp. 774–783, April 1989.

[22] S. S. Panwar, D. Towsley, and J. K. Wolf, "Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service," 1988.

[23] P. P. Bhattacharya and A. Ephremides, "Optimal scheduling of the transmission of messages with strict deadlines," *IEEE Transactions on Automatic Control*, vol. 34, pp. 721–728, July 1989.

[24] K. W. Ross and B. Chen, "Optimal scheduling of interactive and noninteractive traffic in telecommunication systems," *IEEE Transactions on Automatic Control*, vol. 33, pp. 261–267, March 1988.

[25] M. Pinedo, "Stochastic scheduling with release dates and due dates," *Operations Research*, vol. 31, pp. 559–572, May–June 1983.

[26] P. Nain and K. Ross, "Optimal multiplexing of heterogeneous traffic with hard constraint," in *ACM Joint Conference on Computer Performance Measurement, Modeling and Evaluation*, pp. 100–108, 1986.

[27] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models.* Baltimore, Maryland: Johns Hopkins University Press, 1981.

[28] G. H. Golub and C. F. V. Loan, *Matrix Computations.* Johns Hopkins Series in the Mathematical Sciences, Baltimore, Maryland: Johns Hopkins University Press, 1983.

[29] G. W. Stewart, *Introduction to Matrix Computations.* Computer Science and Applied Mathematics, New York: Academic Press, 1973.

[30] L. A. Hageman and D. M. Young, *Applied Iterative Methods.* Computer Science and Applied Mathematics, Academic Press, 1981.

[31] T. C. Oppe, W. D. Joubert, and D. R. Kincaid, "NSPCG user's guide, version 1.0, a package for solving large sparse linear systems by various iterative methods," tech. rep., Center for Numerical Analysis, The University of Texas at Austin, April 1988.

[32] W. J. Stewart, "A comparison of numerical techniques in Markov modeling," *Communications of the ACM*, vol. 21, pp. 144–152, February 1978.

[33] W. J. Stewart, "MARCA: Markov chain analyzer, a software package for Markov modelling," Computer Science Technical Report TR-88-32, North Carolina State University, Raleigh, NC, October 1988.

[34] V. L. Wallace and R. S. Rosenberg, "Markovian models and numerical analysis of computer system behavior," in *Proceedings of the 1966 AFIPS Spring Joint Computer Conference*, (Washington, D.C.), pp. 141–148, 1966.

[35] A. Goyal and S. S. Lavenberg, "Modeling and analysis of computer system availability," *IBM Journal of Research and Development*, vol. 31, pp. 651–664, November 1987.

[36] J. Bernussou, F. L. Gall, and G. Authié, "About some iterative synchronous and asynchronous methods for markov chain distribution computation," in *IFAC 10th Triennial World Congress*, (Munich, FRG), pp. 39–44, 1987.

[37] J. Koury, D. McAllister, and W. Stewart, "Iterative methods for computing stationary distributions of nearly completely decomposable markov chains," *SIAM Journal on Algebraic and Discrete Methods*, vol. 5, pp. 164–186, June 1984.

[38] L. Kaufman, "Solving large sparse linear systems arising in queuing problems," in *Numerical Integration of Differential Equations and Large Linear Systems, Proceedings of Two Workshops held at the University of Bielfeld* (J. Hinze, ed.), (University of Bielfeld), pp. 352–360, Springer-Verlag, Spring 1982.

[39] P. Joly and R. Eymard, "Preconditioned biconjugate gradient methods for numerical reservoir simulation." To appear in *Journal of Computational Physics.*

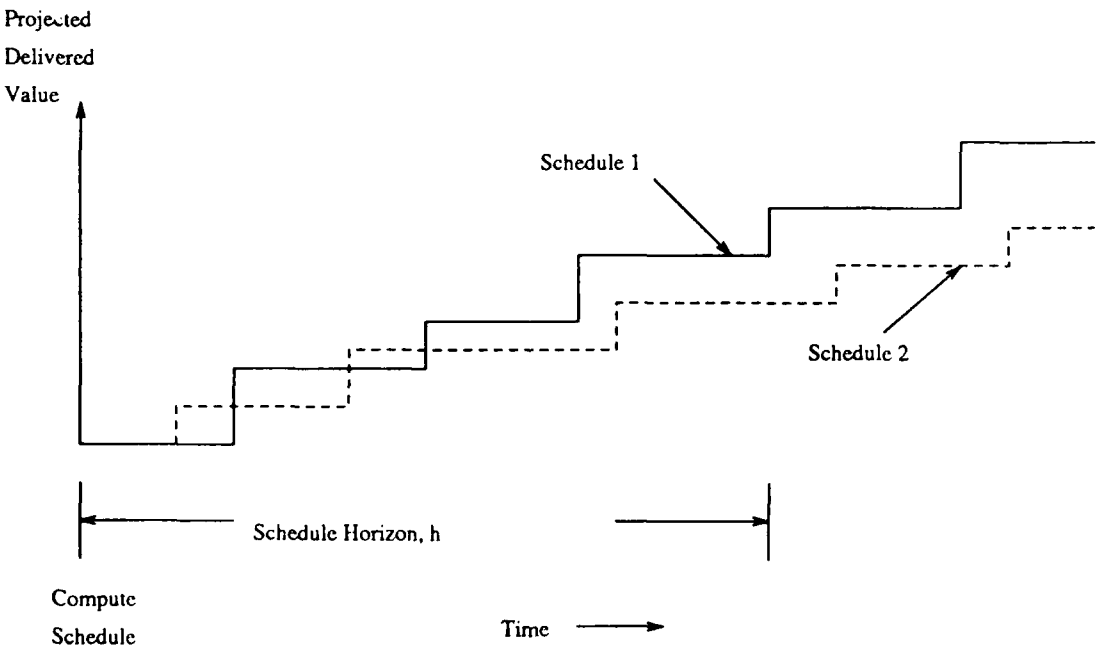


Figure 1: Comparison of alternative schedules

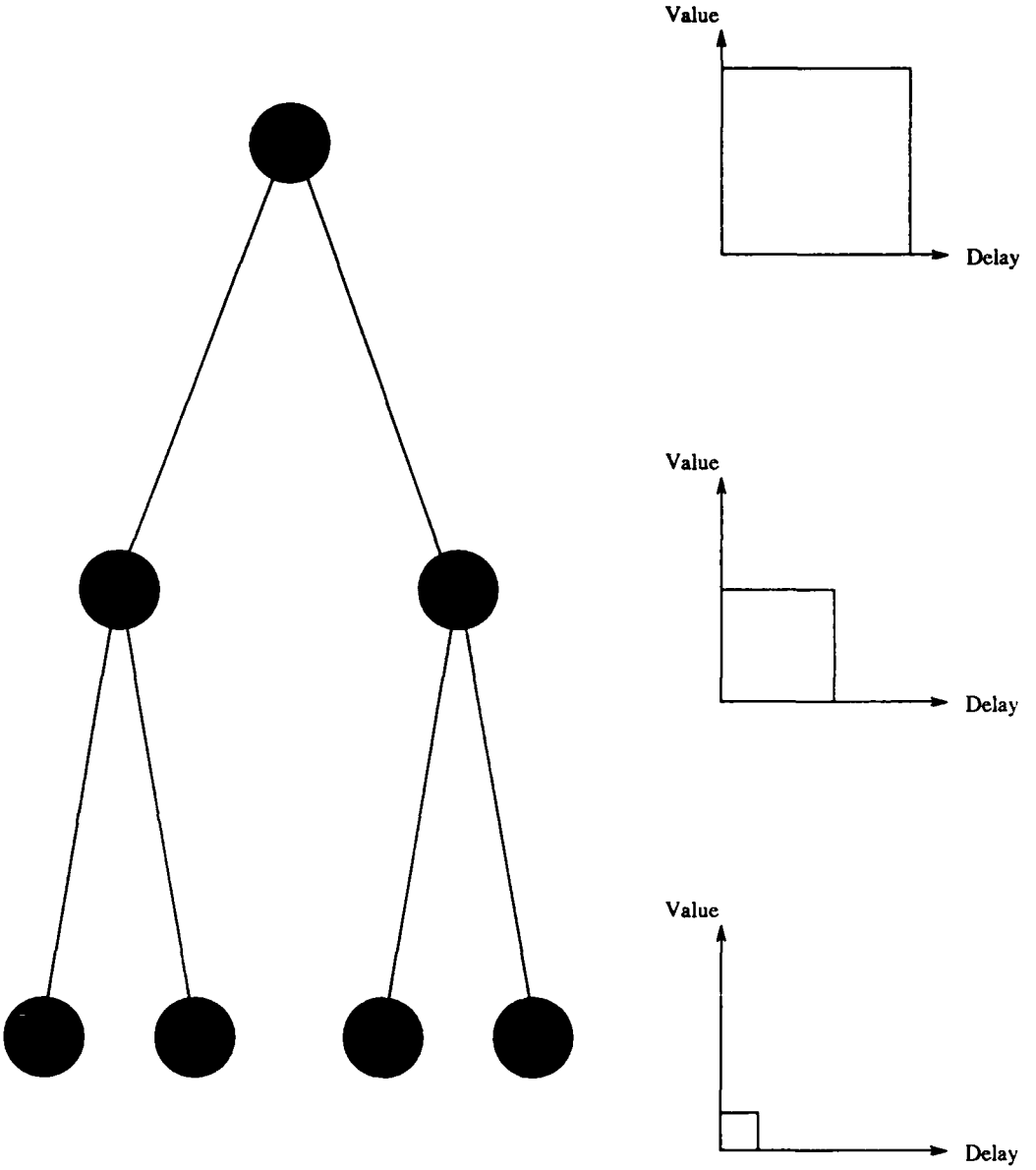
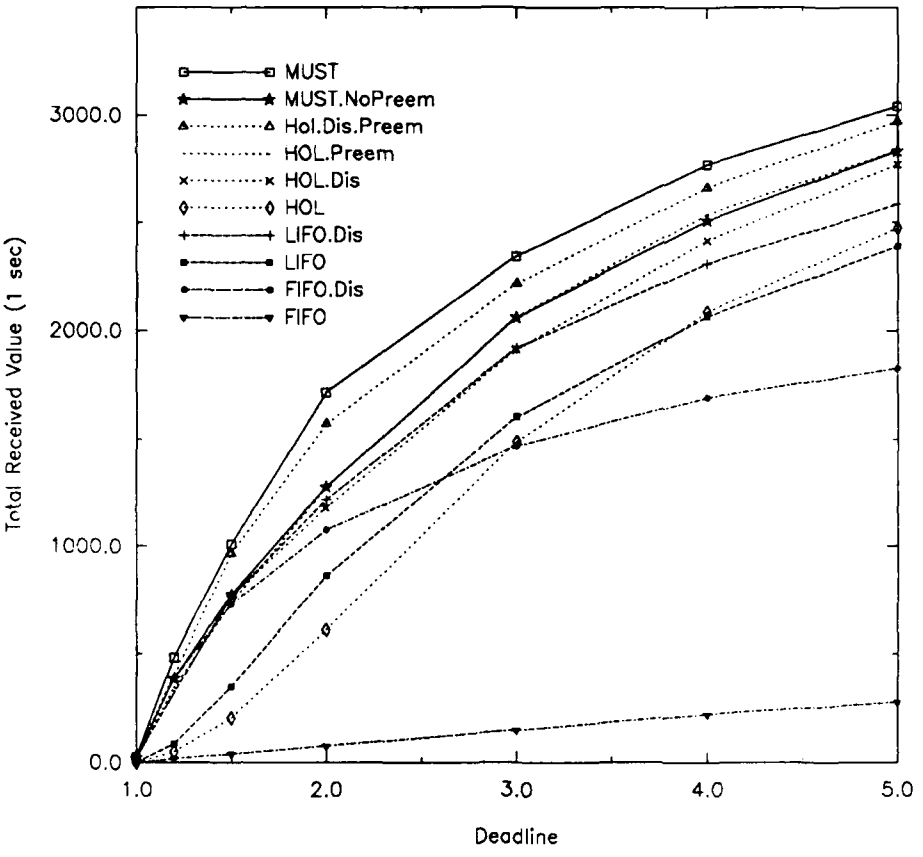


Figure 2: A possible case of value functions by echelon



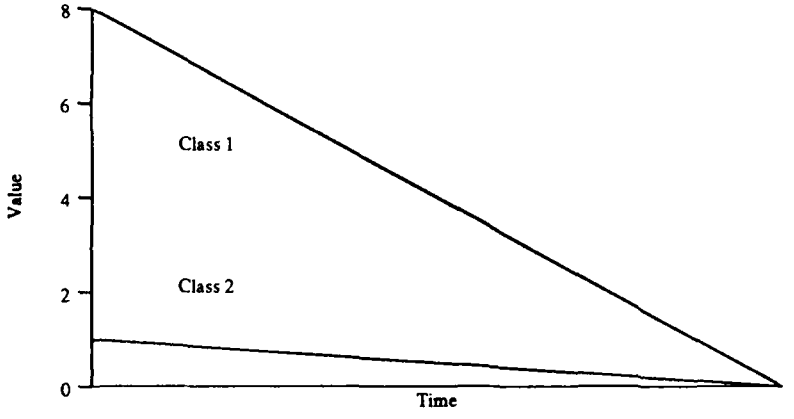


Figure 3: Comparison of protocols where value functions linearly decreases to deadline, varying deadline

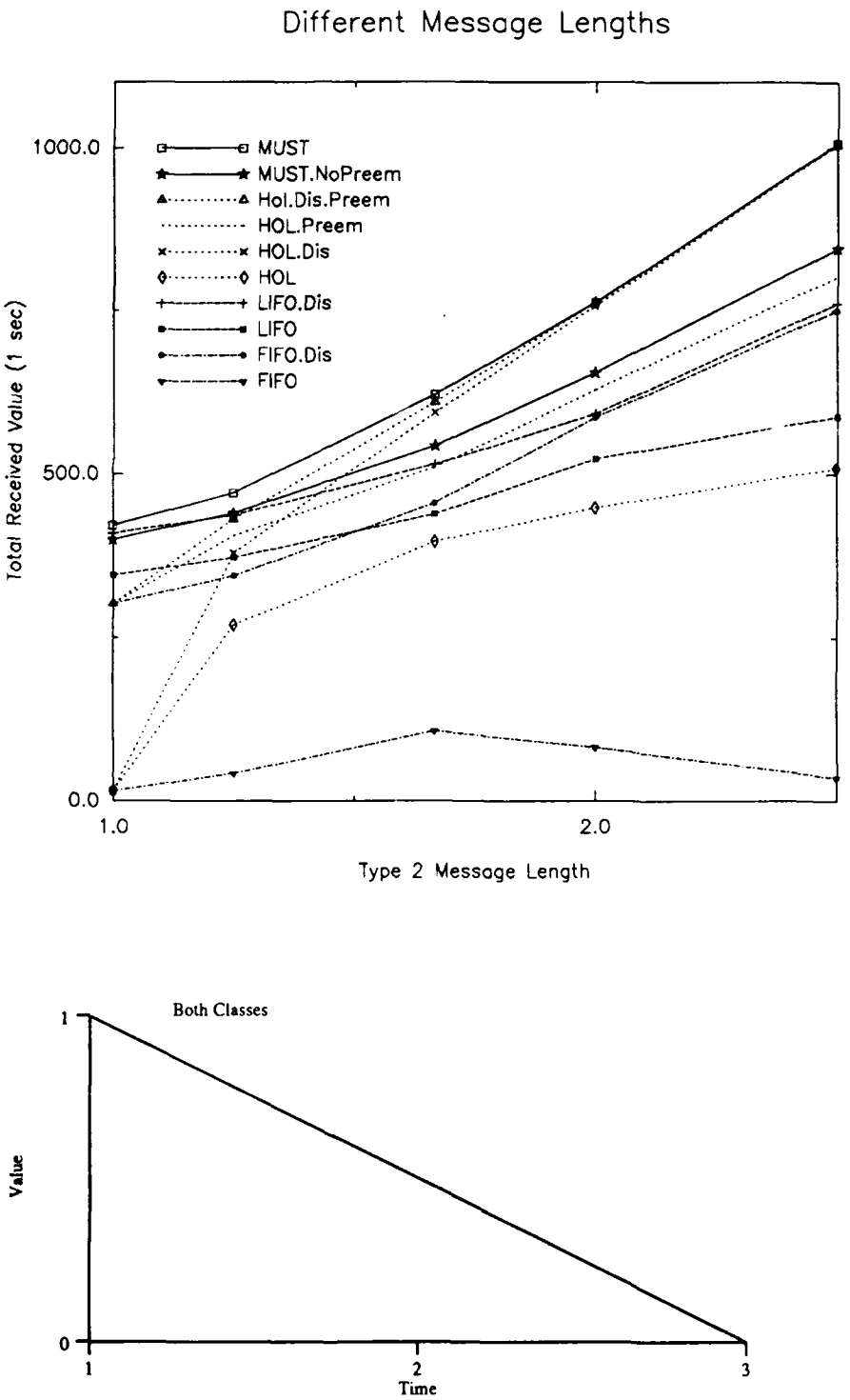
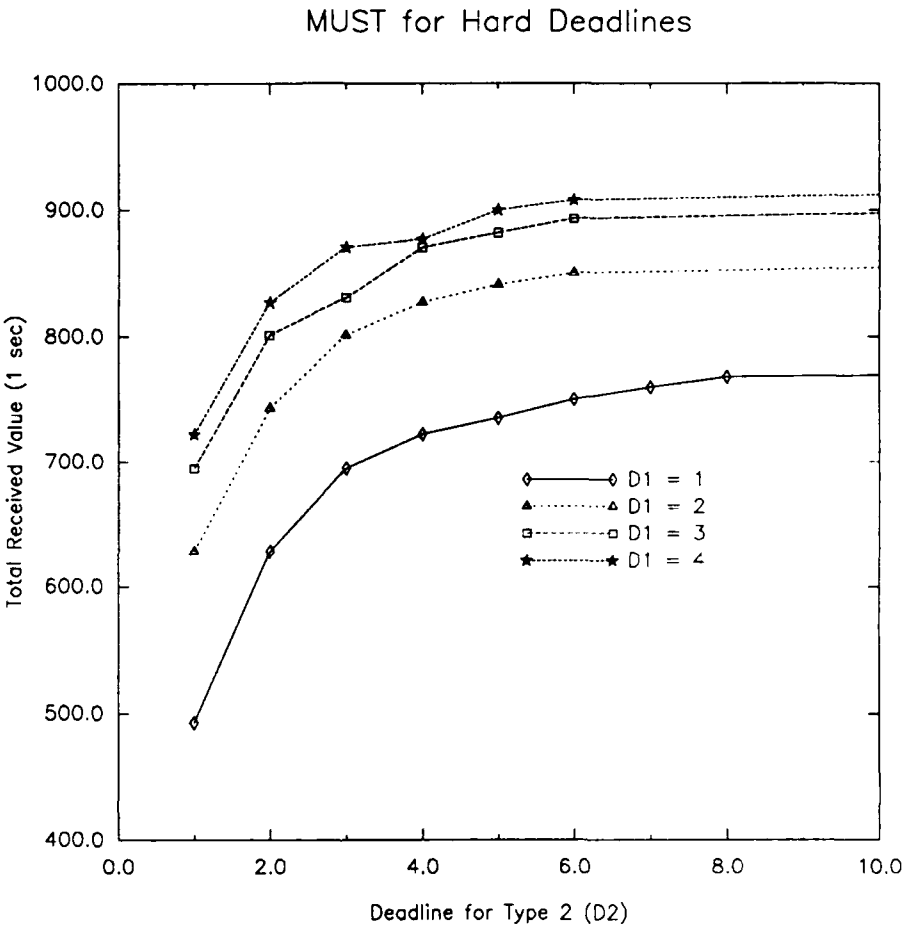


Figure 4: Comparison of protocols where value functions are identical but disparity exists in message lengths



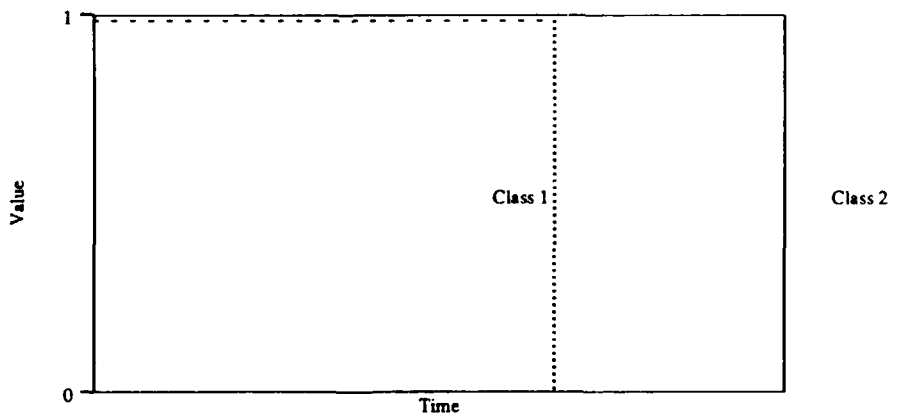
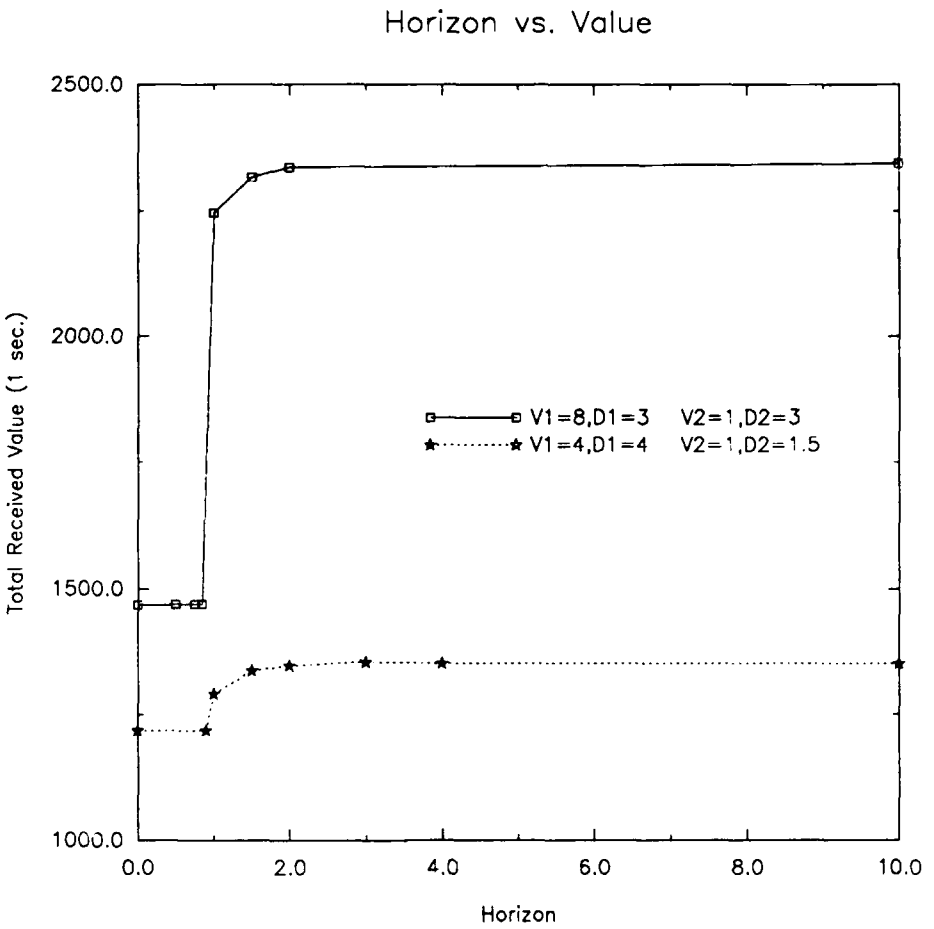


Figure 5: Delivered value for hard deadlines



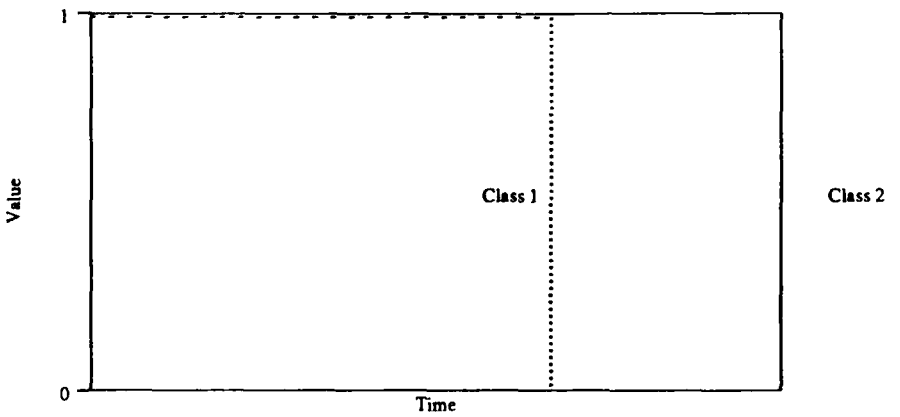


Figure 6: Impact of Finite Horizon

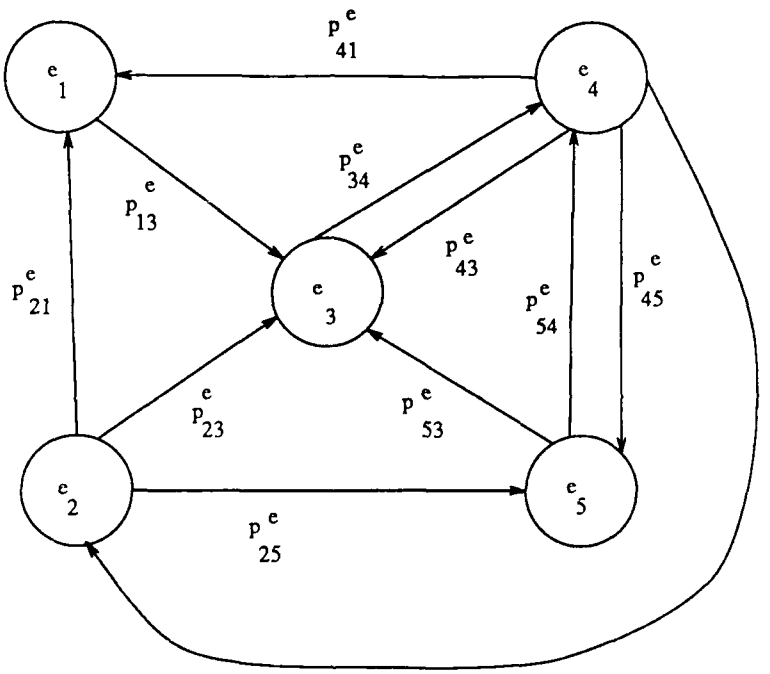


Figure 7: The Random Environment

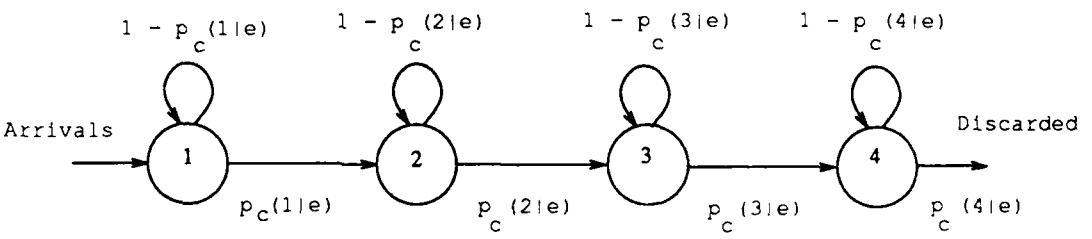


Figure 8: Message Phase Transitions

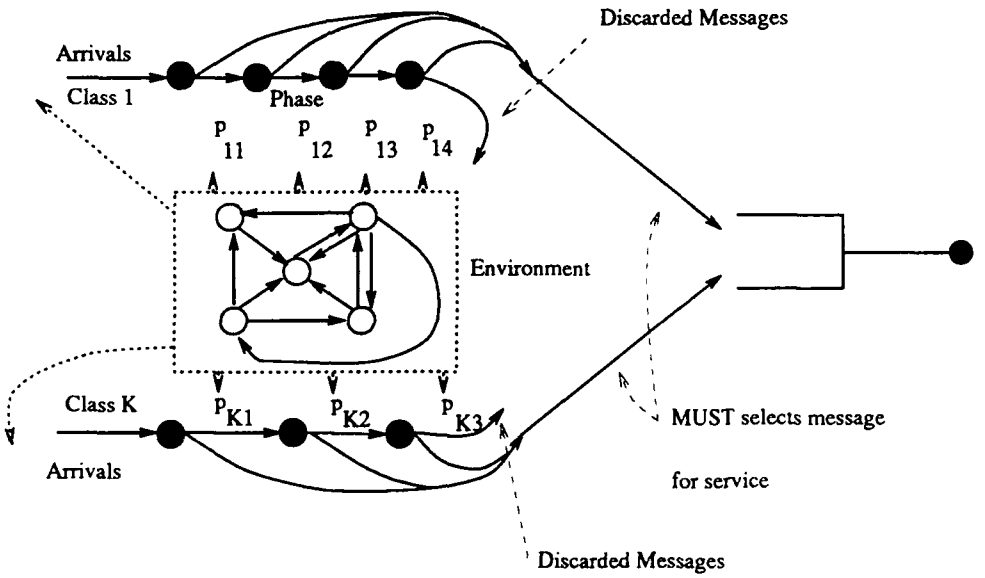


Figure 9: Interaction between Environment and Message Processes

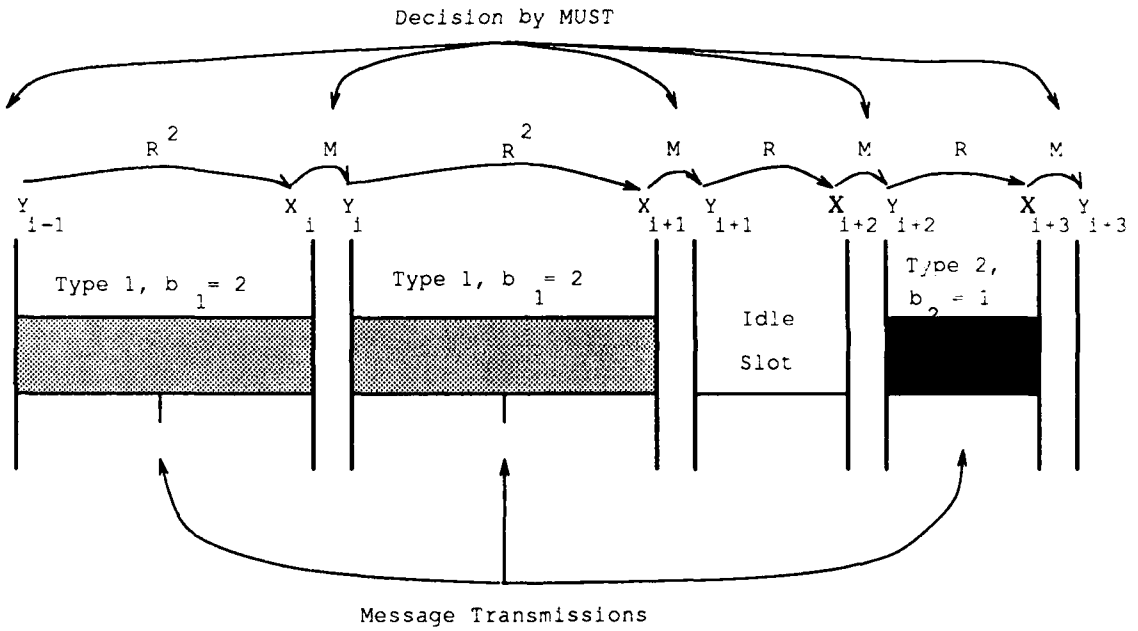


Figure 10: The MUST Decision Points

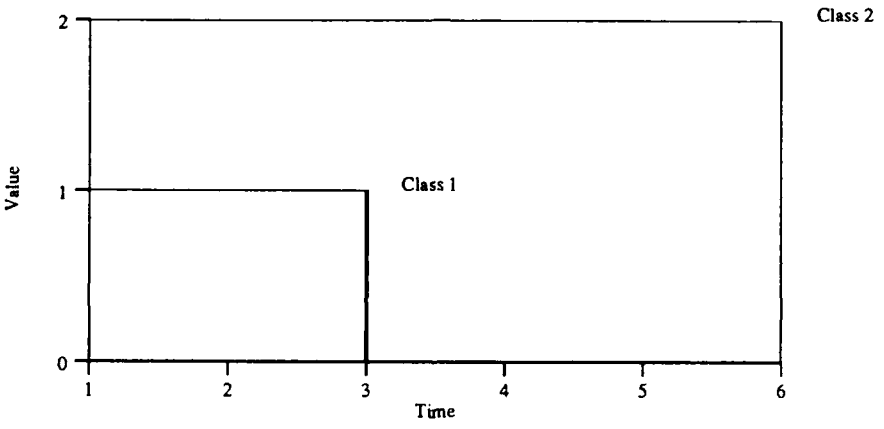


Figure 11: Value functions for case 1

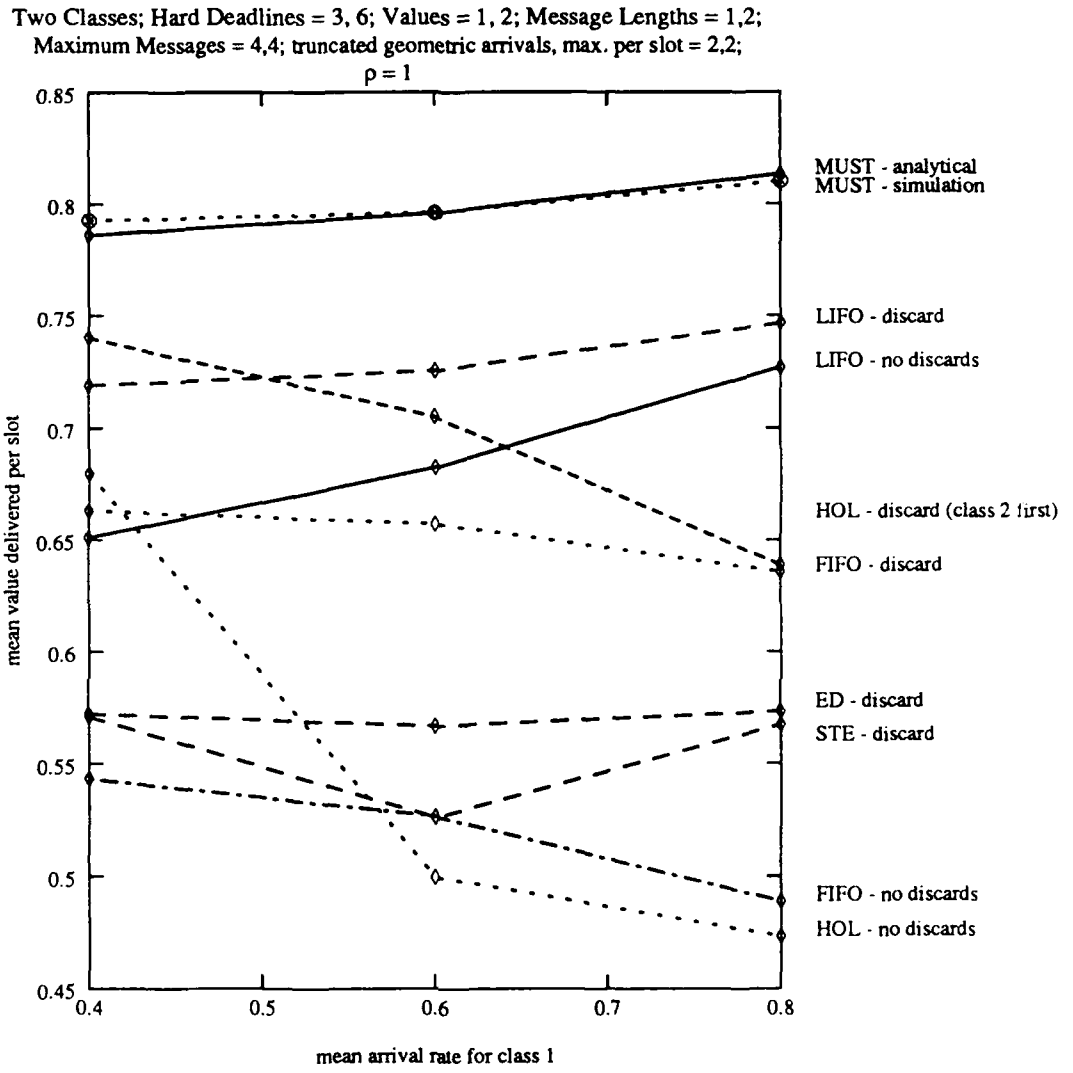


Figure 12: Case 1 - Hard Deadlines

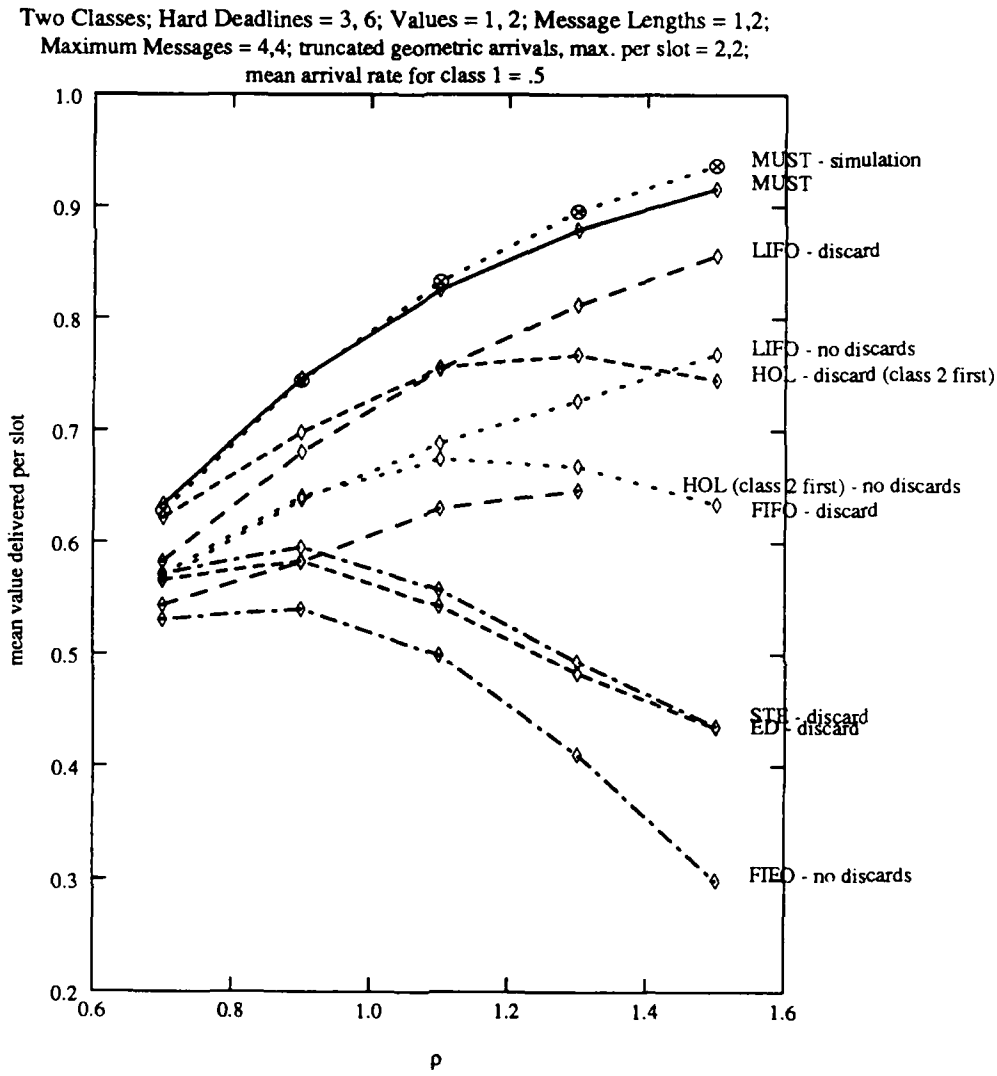


Figure 13: Case 1 - Effect of Increasing Class 2 Traffic

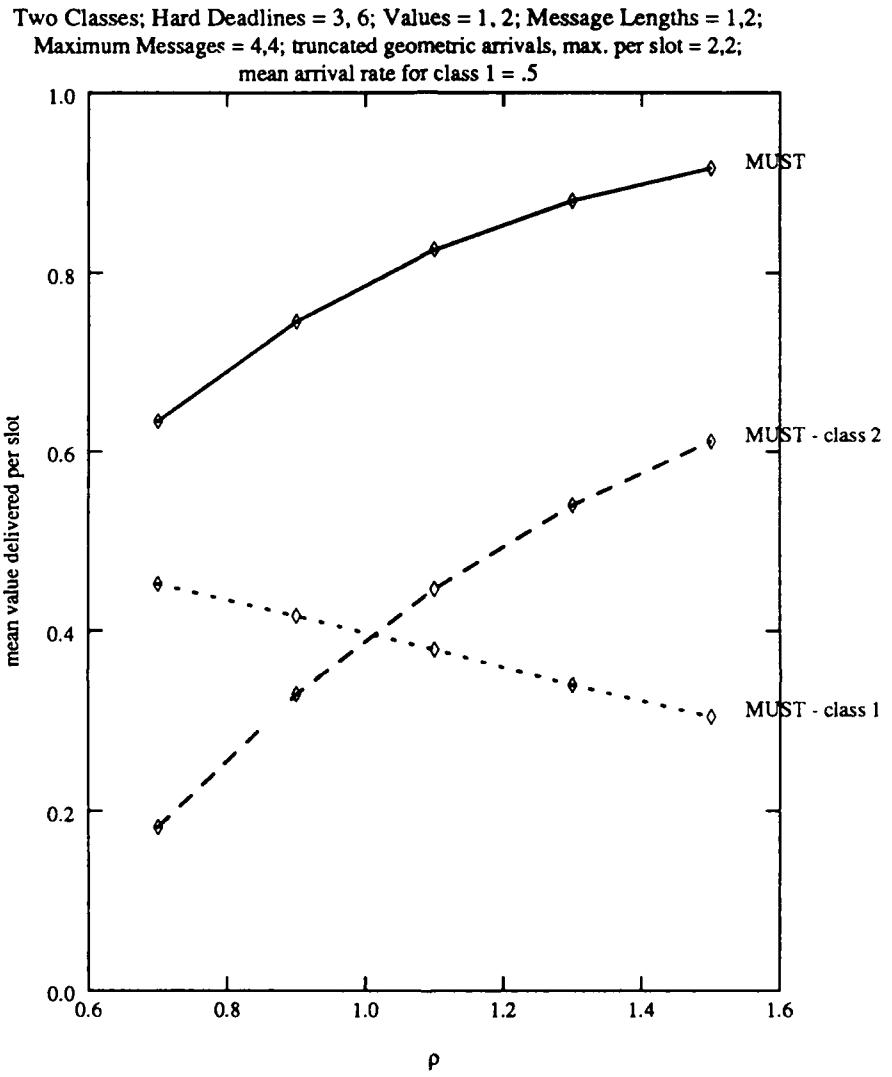


Figure 14: Case 1 - Division of Value Delivered by Class

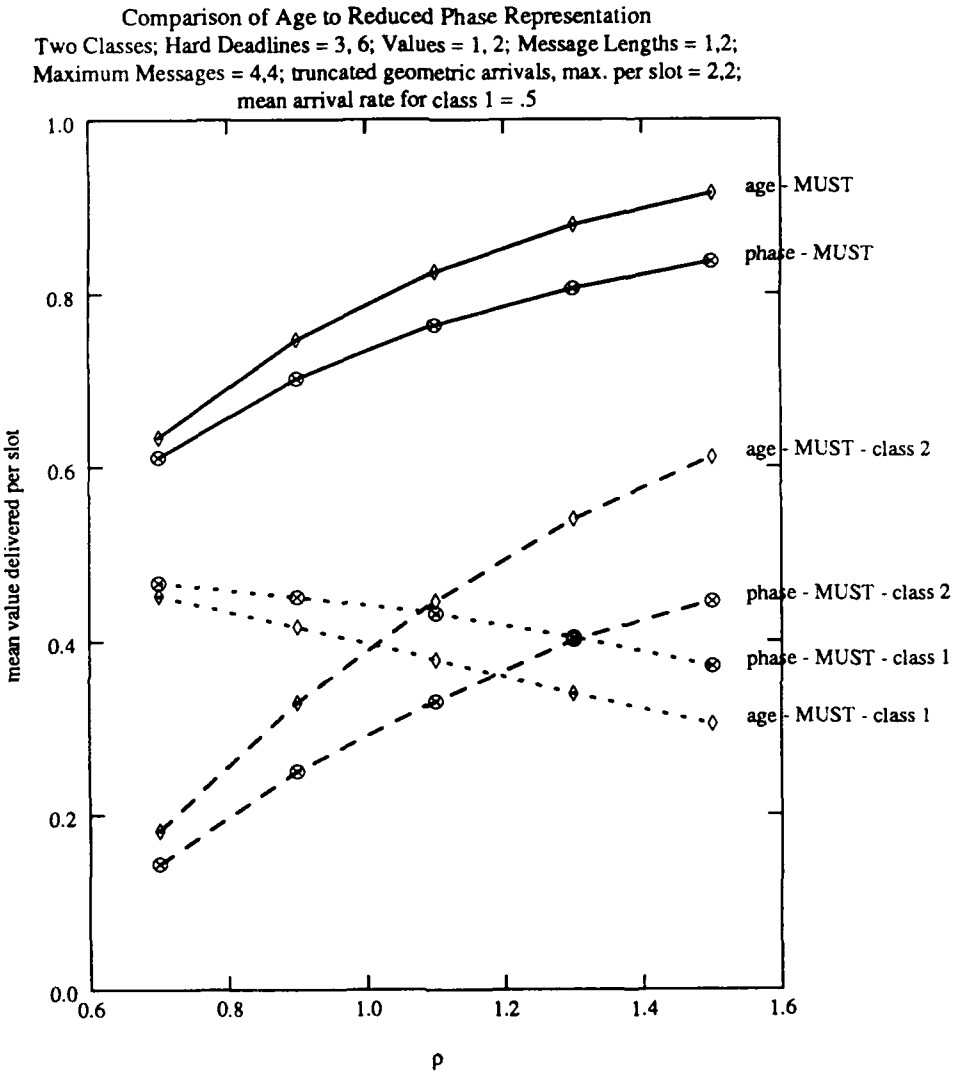


Figure 15: Case 1 - Reduced Phase Representation

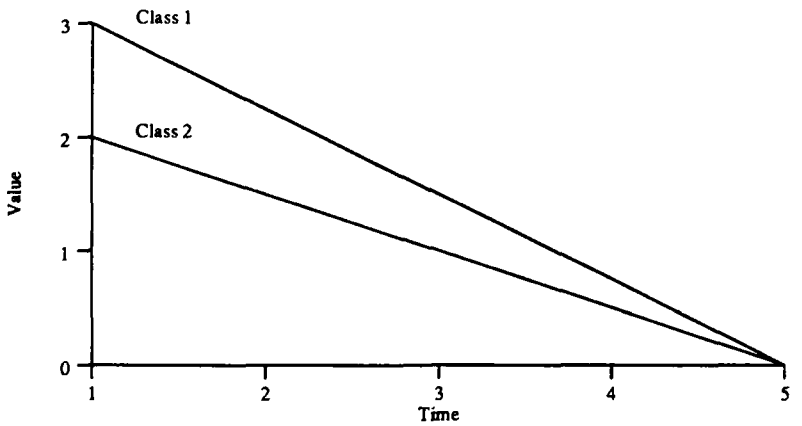


Figure 16: Value Functions for Case 2

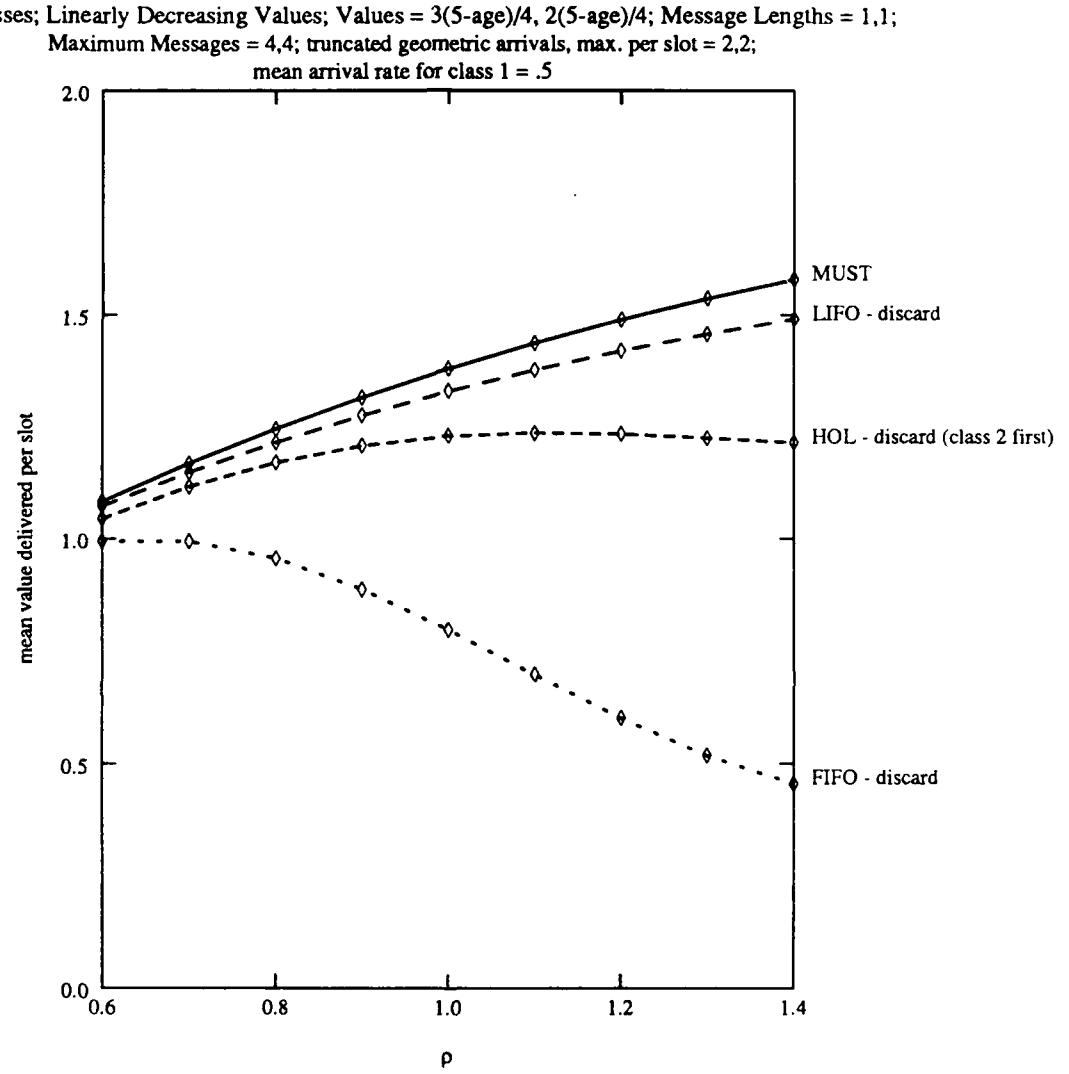


Figure 17: Case 2 - Linearly Decreasing Value Functions

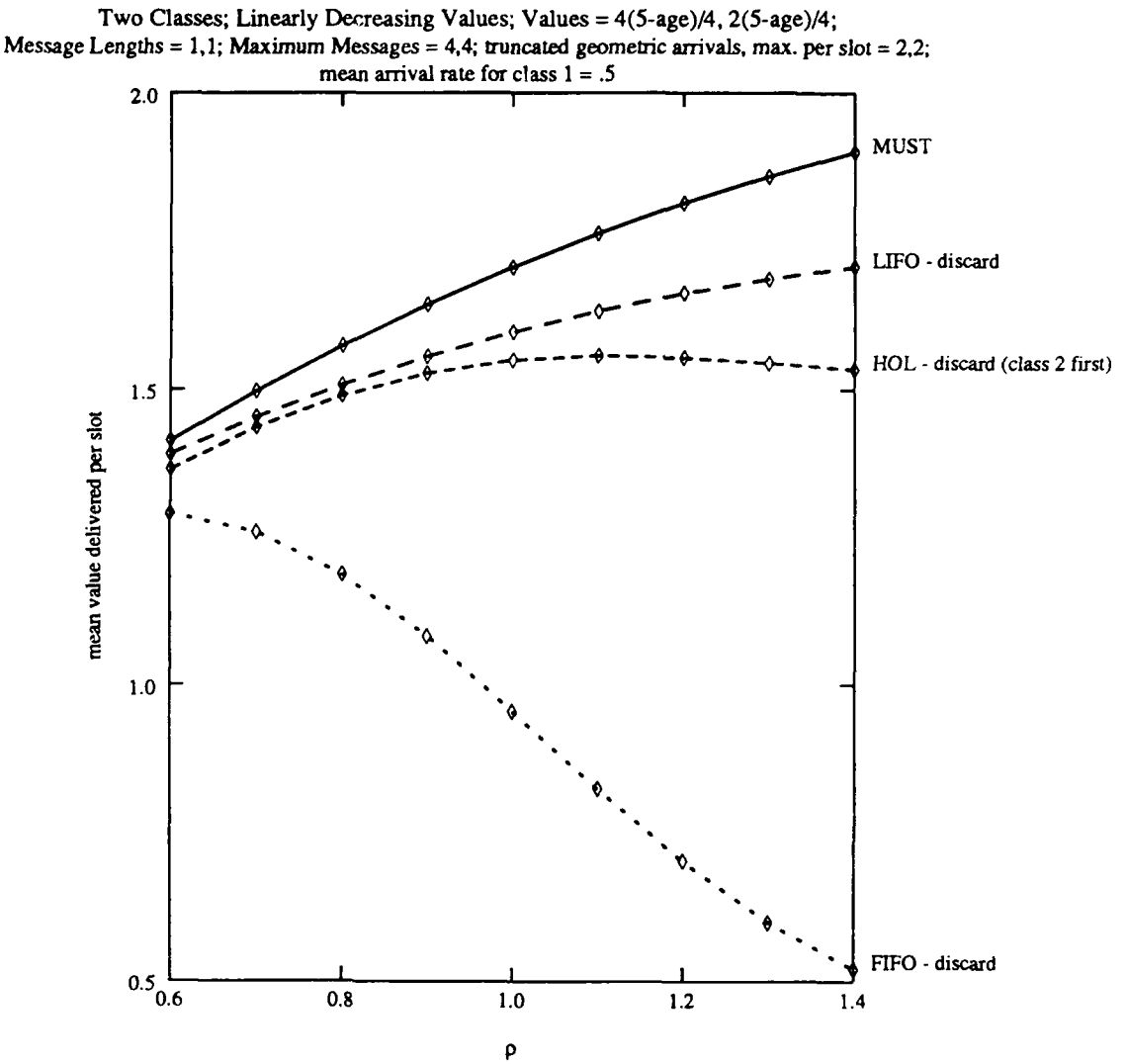


Figure 18: Case 2 - Linearly Decreasing Value Functions

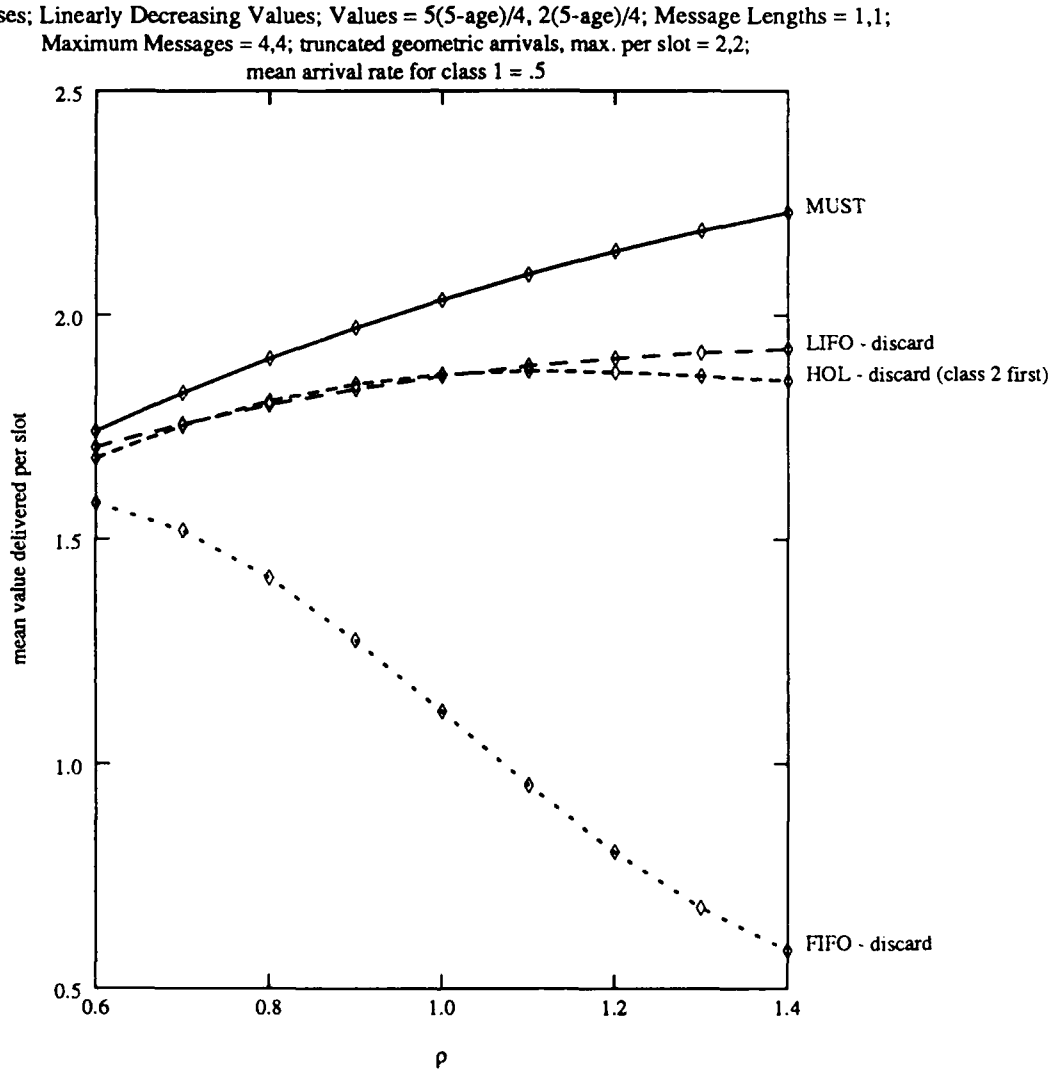


Figure 19: Case 2 - Linearly Decreasing Value Functions

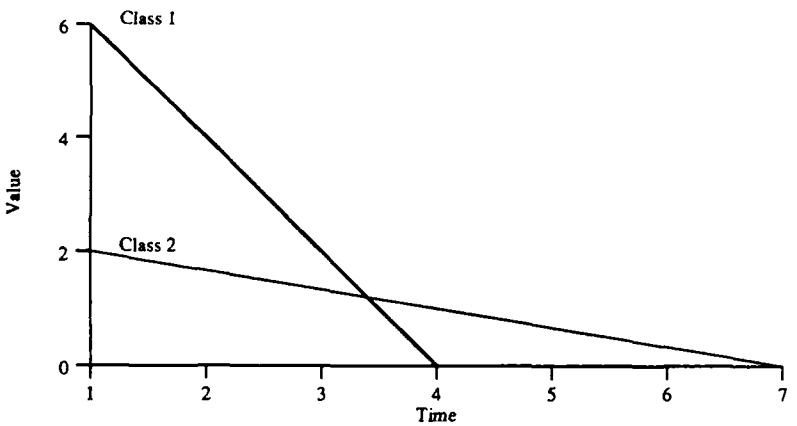


Figure 20: Value functions for case 3

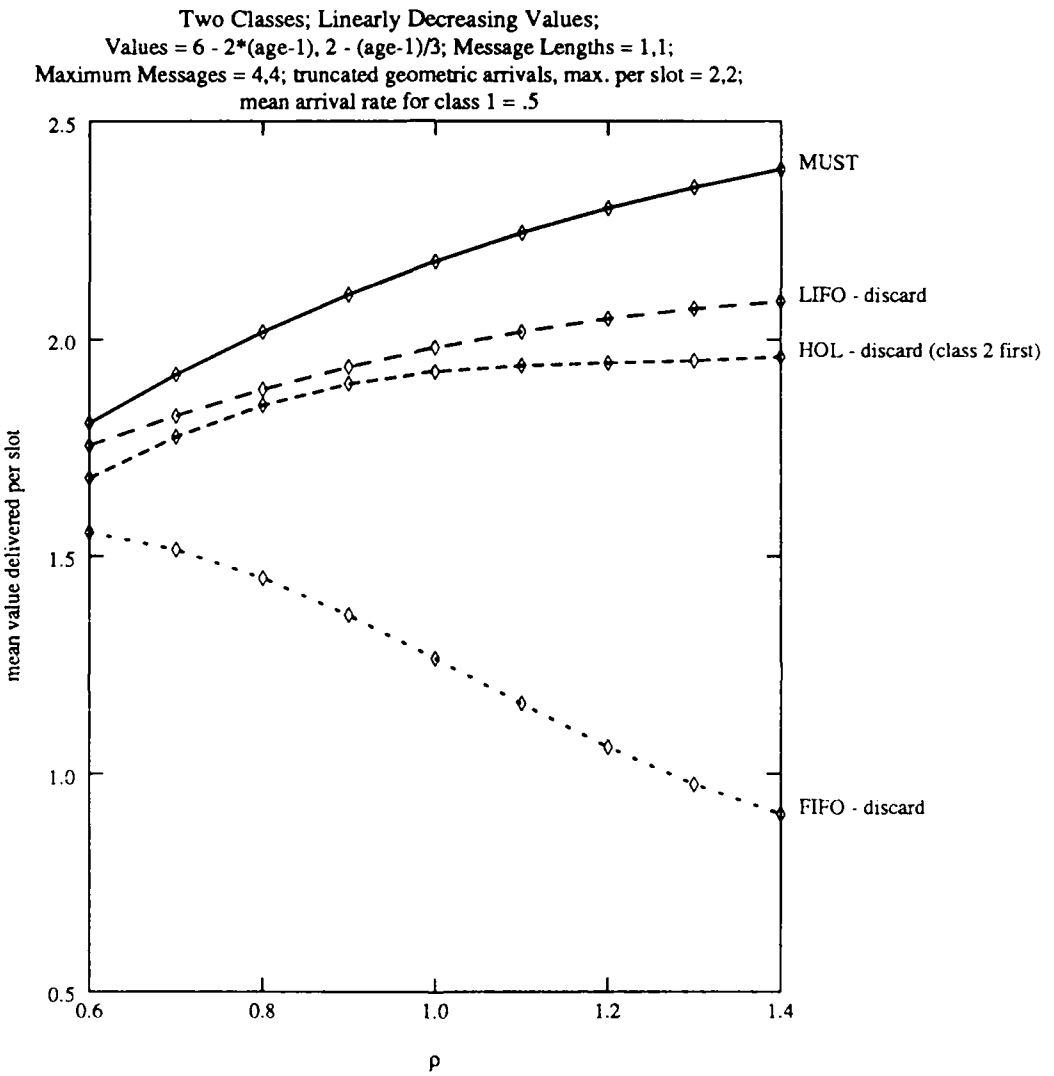


Figure 21: Case 3

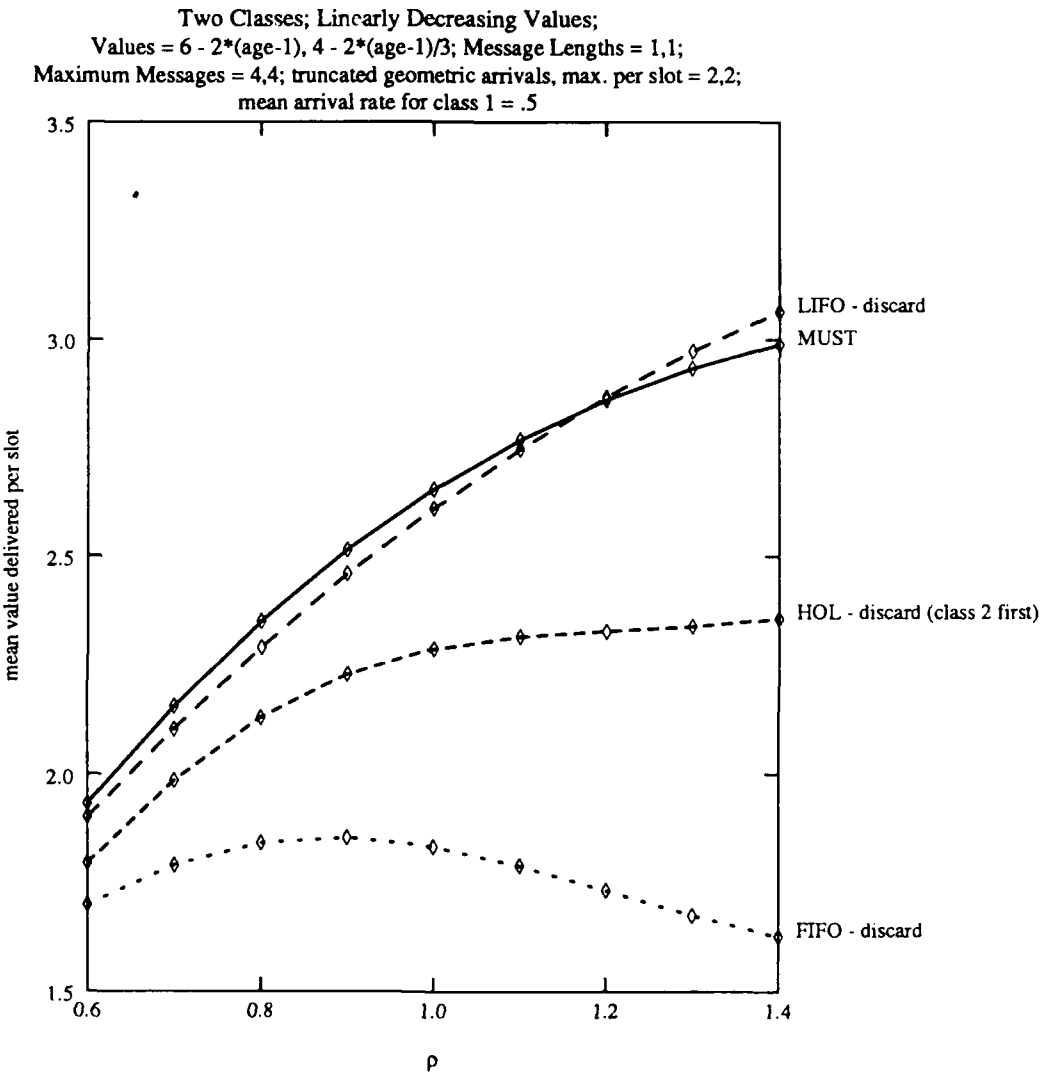


Figure 22: Case 3

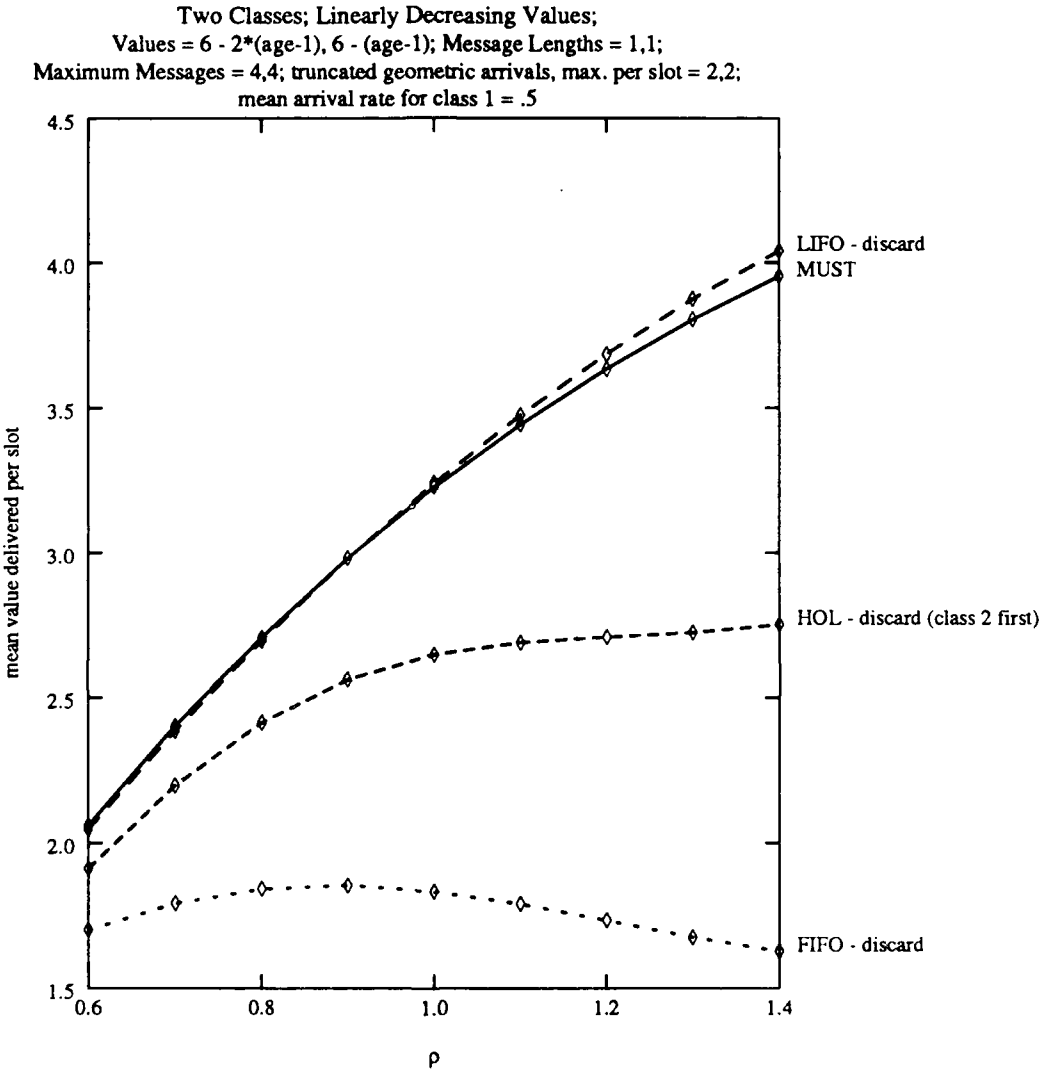


Figure 23: Case 3

APPENDIX II

APPENDIX II

MULTISENSOR DATA FUSION WITH MINIMAL TRANSMISSION FOR TACTICAL SYSTEMS*

J.M. Richardson
Rockwell International Science Center
Thousand Oaks, CA 91360

ABSTRACT

Distributed decision making is a methodology with increasing importance in many applications (e.g., command of naval fleets, battle management of ground forces, anti-submarine warfare, SDI, and many others). An essential aspect of this methodology is the estimation of the state of knowledge (i.e., the conditional probability density in state space) based upon the outputs of a dispersed set of sensors. The current state of knowledge is evolved according to a dynamical model and is continually or intermittently conditioned on all past measurements. An efficient way of handling the estimation process is to equip each sensor station with a certain signal-processing capability and then to transmit bandwidth limited signals to a central fusion station (or several such stations) where the estimation of the most comprehensive state of knowledge is completed. This paper provides a rigorous conceptual basis for accomplishing the above fusion operation efficiently, but in a manner that is exactly equivalent to the state of knowledge obtained directly from all available raw data. The rigorous fusion process is possible only in the case of deterministic dynamical models but with random initial conditions. Two versions of the conditional Fokker-Planck (CFP) methodology giving the evolution of the state of knowledge are considered: (a) a single CFP equation based upon

*This investigation was supported by ONR contract N00014-87-C-0703.

all available sensor data and (b) a family of CFP equations each of which is based upon the sensor data available at the station with which it is associated. A simple fusion formula that relates the solution of (a) to the solutions of (b) and to the unconditional state of knowledge is derived. Practical problems of near-optimal implementation and their impact on the design of the communication system are discussed.

1.0 INTRODUCTION

Distributed decision making is embedded in a stochastic system to be controlled by procedures to be optimized according to some appropriate global criterion. Whatever the latter is and whatever the control procedures (optimal or suboptimal) are, all of the relevant information for carrying out actual control decisions is provided by the current conditional probability density (CPD) in state space (or by some suitable approximate representation of it). In Fig. 1 we present a schematic representation of the above statements in terms of management of a typical defense situation (e.g., naval battle, ground conflict, ASW, SDI, etc.) The part of the block diagram enclosed by the dashed lines includes the process of determining the CPD (i.e., the state of knowledge) from signals produced by a dispersed set of sensor stations. Exterior to the dashed lines we have placed two modules involved in the control decision process, i.e., the distributed controller and human supervision, which are both influenced in an off-line sense by the global criterion of performance. Returning to interior of the space enclosed by the dashed lines we have included communication channels for sending control commands to the defensive system and to the sensor stations. The total communication system is represented by set of small boxes labelled C.

The CPD, updated on dynamics and measurements at all past times, is given by the solution to the conditional Fokker-Planck (CFP) equation. This equation represents the most advanced outer limit of the body of work under the general heading of Kalman[1] filter theory. In fact, all of the results of this body of work can be derived from the CFP equation. An important point to keep in mind is that the concept of optimality (global or local) is irrelevant to the CPD. This entity is "pre-optimal" in the sense that it is the

input into any conceivable decision procedure (e.g., control, state estimation) based upon any conceivable global optimality criterion.

The general objective of this paper is to formulate on a conceptual level the determination of the state of knowledge (i.e., the CPD). A particular objective is deviation of a conceptually rigorous procedure for transmitting fusible signals from dispersed sensor stations to a central fusion station (or several such stations) where the comprehensive state of knowledge based upon all data is computed. This forms the conceptual basis for the use of distributed sensing without overloading the communication system. The avoidance of overload depends, of course, on the choice of approximate representations for the fusible signals.

Although the results of this paper have considerable generality, these results are applied specifically to the problems of single target and multitarget tracking or to the somewhat more general problem of the surveillance of moving objects. In the latter problem we assume that the traditional division into acquisition and tracking is not possible and that these functions are simultaneously handled over a period of time.

In the determination of the CPD, we are concerned with the following aspects: (1) data fusion in the context of a network of sensor stations each of which has some degree of signal processing capability, (2) the handling of general temporal sequences of measurements including time delays, (3) the problem of representing the maneuvering process by a parametric model. We will expand upon these items in the immediately following paragraphs.

The main objective of this paper is the derivation of some results pertaining to the conceptual aspects of data fusion. The problem can be simply stated as the determination of the proper way to combine in a fusion station signals from a set of

sensor stations so that the result is exactly the same as the result that would be obtained if all of the raw data were transmitted directly to the fusion station. In the system under consideration here, the discussion is not limited to a single fusion station. Also questions concerning system reliability or even survivability encourage the use of a multitude of fusion stations, appropriately dispersed, to achieve an adequate degree of redundancy. Some or all of these fusion stations may be combined with sensor stations.

The problem of fusion in the context of time-independent states (i.e., states that are either constant in time, but unknown, or states in a dynamic context that are considered at a specified instant of time) has been studied by a number of investigators. In the relatively restricted area of multitarget tracking we can cite Mori, Chong, Tse, and Wishner,[2] Chong and Mori,[3] and others. Useful review papers have been published by Bar-Shalom[4] and Reid.[5] In the dynamical case, involving single or many targets, results on fusion are rather sparse, as far as we know, because of fundamental difficulties pointed out by Chong and Mori. The approach to fusion in the dynamical case discussed in this paper circumvents these difficulties rather than dealing with them directly.

We turn now to the second aspect of the CPD determination problem, namely the handling of a general sequence of measurements. In the problem formulated in the next section we explicitly consider measurements made on a continuous sequence of times. But we also wish to consider an intermittent set of measurements made at discrete times that may be periodic or aperiodic. Also we want to be able to handle the case in which measurements at different stations are mutually asynchronous. It is important to be able to deal with various kinds of time delays, both those that are known a priori and those that are not. The latter type of time delay occurs in the measurements of targets whose ranges relative to each sensor station are not known a priori and must be inferred, if at all, in a later phase of the total inference process.

The third aspect pertains to the fact that if the dynamical model involves noise (either in the form of a natural perturbation or a control vector of unknown temporal structure yielding random maneuvering), it is then impossible to fuse the CPDs produced at different stations to produce correctly the total CPD corresponding to the set of raw data obtained at all stations. We consider a so-called "parametric" maneuvering model in which the control vector is related to the solution of a set of deterministic differential equations with random initial conditions. This is not nearly as restrictive as it might appear at first sight since this type of model corresponds to a bundle of deterministic trajectories whose probabilities are continually and/or intermittently revised by the conditioning process. Thus, on a gross scale the maximum probability trajectory will not, in general, conform to the noiseless dynamical model.

2.0 STATEMENT OF PROBLEM

Let us consider the determination of the optimal estimate based upon all data and then later generalize the formulation to embrace the problem of fusion. The system of interest is assumed to be a single object (e.g., a missile in an exo-atmospheric environment or a submarine in the ocean). We make the important restrictive assumption that the object is at most subject to "parametric maneuvering," namely its dynamics (including "parametric maneuvering") is described by deterministic differential equations with random initial conditions.

We accordingly assume that the dynamical and measurement models are represented by

$$\dot{x}(t) = f(x(t),t) \quad , \tag{2.1}$$

$$y(t) = g(x(t),t) + \nu(t) \quad , \tag{2.2}$$

where

$x(t)$ = state of the object at time t (an m-dimensional vector) including maneuvering parameters,

$f(x(t),t)$ = an m-dimensional vector function of $x(t)$ and t giving the rate of change of $x(t)$ in the absence of noise,

$y(t)$ = the set of measurements at time t (an n-dimensional vector),

$g(x(t),t)$ = an n-dimensional vector function of $x(t)$ and t,

$\nu(t)$ = an n-dimensional Gaussian random process.

This Gaussian random process has the a priori statistical properties

$$E\nu(t) = 0 \tag{2.3}$$

$$E\nu(t)\nu(t')^T = R_\nu \delta(t-t') \tag{2.4}$$

where R_ν is a constant symmetric, positive-definite n×n matrix. We further assume that $\nu(t)$ and $x(t')$ are statistically independent when $t > t'$.

Now, the problem at hand involves the consideration of separate surveillance operations carried on by two sensor stations, each with its own independent sensor system. Denoting the measurement vectors for the two sensor systems by $y_1 = y_1(t)$ and $y_2 = y_2(t)$, the processing in the two stations can be represented by the two measurement models.

$$y_i(t) = g_i(x(t),t) + \nu_i(t), \; i = 1,2 \quad , \tag{2.5}$$

where the $\nu_i(t)$ are statistically independent Gaussian random processes with

$$E\nu_i(t) = 0, \; E\nu_i(t)\nu_i(t')^T = R_{\nu_i}\delta(t-t'), \; i = 1,2. \tag{2.6}$$

To provide some perspective on the later mathematical development a quasi-rigorous discussion of the fusion problem is presented here. The measurement vectors $y_1(t)$ and $y_2(t)$ represent the measurements made at time t at sensor stations 1 and 2, respectively, and the state vector x(t) gives the state of an object at time t (assuming for the moment that only one object is present). y_1^t, y_2^t, and x^t denote the complete time-histories of $y_1(t)$, $y_2(t)$, and x(t) in the interval (0,t). In the dynamical case the assumption of the conditional independence of y_1^t and y_2^t must be expressed in the form

$$P(y_1^t, y_2^t | x^t) = P(y_1^t | x^t) P(y_2^t | x^t) \quad , \tag{2.7}$$

as pointed out by Chong and Mori.[3] Because of the assumption that the dynamical model is deterministic, but with random initial conditions, it follows that the state x(t') at any

time t' determines the entire history of x, and in particular x(t) implies x^t. In this case (2.7) reduces to

$$P(y_1^t, y_2^t | x(t)) = P(y_1^t | x(t)) P(y_2^t | x(t)). \tag{2.8}$$

Some straightforward manipulations yield the relation

$$P(x(t) | y_1^t, y_2^t) = a(t) P(x(t) | y_1^t) P(x(t) | y_2^t) P(x(t))^{-1} \quad , \tag{2.9}$$

where a(t) is a normalization constant. To make the above results rigorous the abstract mathematical apparatus of the theory of stochastic processes (see, for example, the well-known treatise by Doob[6]) must be used. However, this problem can be handled simply and rigorously within the mathematical framework of the CFP equation. This latter approach has the added advantage of expressing the results in terms of the notation of the CFP equation.

In dealing with the simple, rigorous proof using the CFP, another representation of the PD, namely P(x,t|S), which is the PD (conditioned on the set S) at time t in the neighborhood of the fixed but arbitrary reference point x, must be considered. The set S could be the null set ϕ, y_1^t, y_2^t, or the combination of y_1^t and y_2^t. To further simplify notation we will put $P(x,t|\phi) \equiv P(x,t) \equiv P_o(x,t)$; $P(x,t|y_i^t) = P_i(x,t)$, i = 1,2; and $P(x,t|y_1^t, y_2^t) = P_{12}(x,t)$. The PD conditioned on all measurements in the time interval (0,t), i.e., $P_{12}(x,t)$, evolves in time in accordance with the CFP equation.

$$\frac{\partial P_{12}}{\partial t} + \frac{\partial}{\partial x}^T f P_{12} + \frac{1}{2} (H - \lambda) P_{12} = 0 \ , \tag{2.10}$$

where

$f = f(x,t)$ = function defined in (2.1),

$$H = H(y(t),x,t) = (y(t) - g(x,t))^T R_v^{-1} (y(t) - g(x,t)) \quad , \tag{2.11}$$

$\lambda = \lambda(t)$ = normalization parameter determined by the relation

$$\lambda = \int dx HP \quad . \tag{2.12}$$

The form of H in Eq. (2.6) is based on the assumption that y(t) is bandwidth limited. Thus, here y(t) is an extremely improbable sample of the random process (2.2). On the other hand, a random sample of this process is almost surely a signal with infinite bandwidth. The latter situation leads to a different form of the CFP equation derived by Kushner[7] and Bucy.[8] The CFP equation (2.10) is to be solved with a specified initial condition representing our a priori (i.e., for $t \leq 0$) statistical knowledge.

Now, the problem at hand involves the consideration of separate surveillance operations carried on by two sensor stations. Denoting the measurement vectors for the two sensor systems by $y_1 = y_1(t)$ and $y_2 = y_2(t)$, the processing in the two stations can be represented by the two CFP equations

$$\frac{\partial P_i}{\partial t} + \frac{\partial^T}{\partial x} f P_i + \frac{1}{2} (H_i - \lambda_i) P_i = 0 \ , \ i - 1,2, \tag{2.13}$$

where

$$H_i = (y_i(t) - g_i(x,t))^T R_{\nu_i} (y_i(t) - g_i(x,t) \quad , \tag{2.14}$$

and where λ_i is determined by the normalization condition. The unconditional Fokker-Planck (UFP) equation is given by the simple expression

$$\frac{\partial P_o}{\partial t} + \frac{\partial^T}{\partial x} f\, P_o = 0. \tag{2.15}$$

The initial conditions for all of the above partial differential equations are assumed to be the same, i.e., $P_o(x,0) = P_1(x,0) = P_2(x,0) = P_{12}(x,0)$.

The question of possible fusion reduces to the question: at a time t can P_o, P_1, and P_2 be combined in some way to yield P_{12}? In other words, can relatively simple signals produced by the local processors (at stations 1 and 2 using $y_1(t)$ and $y_2(t)$ as inputs) be combined in such a way that the results are equivalent in information content to transmitting the full set of raw data composed of $y_1(t)$ and $y_2(t)$ continuously to the central station and then solving the CFP equation (2.10)? The answer is "yes" and the procedure for doing so is outlined in the next section.

3.0 SOLUTION

In preparation, the various Fokker-Planck equations (2.10), (2.13), and (2.15) are written in slightly different forms by using the identity

$$\left(\frac{\partial}{\partial t} + \frac{\partial^T}{\partial x} f\right)(\cdot) \tag{3.1}$$
$$= \left(\frac{D}{Dt} + \eta\right)(\cdot) \quad ,$$

where

$$\frac{D}{Dt} \triangleq \frac{\partial}{\partial t} + f^T \frac{\partial}{\partial x} \quad , \tag{3.2}$$

and

$$\eta \triangleq \left(\frac{\partial}{\partial x}^T f\right) \quad . \tag{3.3}$$

The operator defined by (3.2) is analogous to the hydrodynamical total derivative. It represents the rate of change of the operand (i.e., P_{12}, P_1, P_2, or P_o) as perceived by an observer moving through the single-object state space with an instantaneous velocity f(x,t) when he is momentarily at the point x. The quantity η defined by (3.3) represents the rate of expansion of an infinitesimal volume element whose boundaries move with the local velocity f(x,t). The vanishing of η is a necessary condition for the dynamics to be Hamiltonian.

With all of the assumptions and definitions discussed above the various Fokker-Planck equations can be rewritten in the forms:

$$\frac{D}{Dt} P_{12} + \eta P_{12} + \frac{1}{2} (H_1 + H_2 - \lambda) P_{12} = 0 \quad , \tag{2.10a}$$

$$\frac{D}{Dt} P_i + \eta P_i + \frac{1}{2} (H_i - \lambda_i) P_i = 0, \quad i = 1,2, \tag{2.13a}$$

$$\frac{D}{Dt} P_o + \eta P_o = 0 \quad . \tag{2.15a}$$

To reiterate, (2.10a) is conditioned on y_1 and y_2, (2.13a) is conditioned on y_i (i = 1,2), and (2.15a) is not conditioned at all. All of the PDs are subject to the same initial conditions as we have already stated.

We now consider a new quantity Q = Q(x,t) defined by the relation

$$Q = P_1 P_2 P_o^{-1} \quad . \tag{3.4}$$

It is simple to show that Q satisfies the initial condition common to the previously defined PDs. We will now show that Q satisfies (3.6) below. We first obtain

$$\begin{aligned} \frac{D}{Dt} Q &= \frac{D}{Dt} P_1 P_2 P_o^{-1} \\ &= P_2 P_o^{-1} \frac{D}{Dt} P_1 + P_1 P_o^{-1} \frac{D}{Dt} P_2 - P_1 P_2 P_o^{-2} \frac{D}{Dt} P_o \quad , \end{aligned} \tag{3.5}$$

using the fact that D/Dt behaves like an ordinary derivative. Substituting DP_i/Dt (i = 0,1,2) obtained from (2.13a) and (3.14a) we obtain

$$\begin{aligned} \frac{D}{Dt} Q = &-P_2 P_o^{-1} \left(\eta + \frac{1}{2} (H_1 - \lambda_1)\right) P_1 \\ &-P_1 P_o^{-1} \left(\eta + \frac{1}{2} (H_2 - \lambda_2)\right) P_2 \\ &+P_1 P_2 P_o^{-2} \eta P_o \end{aligned}$$

$$= -P_1 P_2 P_o^{-1} \left(\eta + \frac{1}{2} (H_1 + H_2 - \lambda_1 - \lambda_2)\right) \quad ,$$

or

$$\frac{D}{Dt} Q + \eta Q + \frac{1}{2} (H_1 + H_2 - \lambda_1 - \lambda_2) Q = 0 . \tag{3.6}$$

Introducing new dependent variables defined by

$$P = R\exp \left(\frac{1}{2} \int_0^t dt' \, \lambda(t')\right) , \tag{3.7}$$

$$Q = S\exp \left(\frac{1}{2} \int_0^t dt' (\lambda_1(t') + \lambda_2(t'))\right) , \tag{3.8}$$

we can write Eqs. (2.10a) and (3.6) in the forms

$$\frac{D}{Dt} R + \eta R + \frac{1}{2} (H_1 + H_2) R = 0 \quad , \tag{3.9}$$

$$\frac{D}{Dt} S + \eta S + \frac{1}{2} (H_1 + H_2) S = 0 . \tag{3.10}$$

An inspection of (3.7) and (3.8) shows that, since P and Q have the same initial conditions, R and S must also have the same initial conditions. The fact that R and S satisfy the same differential equation (first order in time differentiation) with the same initial conditions implies that

$$R = S, \text{ all } t, \tag{3.11}$$

or, equivalently

$$P = Q\exp\left(\frac{1}{2}\int_0^t dt'(\lambda(t') - \lambda_1(t') - \lambda_2(t'))\right) , \tag{3.12}$$

i.e., P is proportional to Q with a proportionality factor dependent only on time.

The result (3.12) can be written in the form

$$P_{12} = aP_1P_2P_0^{-1} , \tag{3.13}$$

where the quantity $a = a(t)$ is a normalization factor whose value can be obtained by requiring that the r.h. side of (3.11) be normalized or by use of the following equation

$$\log a(t) = \frac{1}{2}\int_0^t dt' \left(\lambda(t') - \lambda_1(t') - \lambda_2(t')\right) , \tag{3.14}$$

a result obtained easily by comparison of (3.12) with (3.13).

Using the original notation, Eq. (3.13) can be re-expressed in the more explicit form given below

$$P(x,t|y_1^t,y_2^t) = a(t)P(x,t|y_1^t)P(x,t|y_2^t)P(x,t)^{-1} . \tag{3.15}$$

This result implies that if two separate sensor stations carry out separate surveillance of a parametrically maneuvering missile, each yielding a CPD $P(x,t|y_i^t)$, $i = 1,2$, at a specified time t, then those outputs can be fused in accordance with (3.13) to yield the CPD $P(x,t|y_1^t,y_2^t)$ that could have been obtained from surveillance at any central station using all measured data in the time interval.

4.0 SUPPLEMENTARY REMARKS

In this section we will discuss a number of generalizations and specializations of the above treatment. A partial list is represented by the following paragraphs.

a. In the above treatment R_μ and R_ν were assumed to be constants independent of the state x, but perhaps dependent upon time. There are many situations in which R_μ and/or R_ν are dependent upon x (e.g., the random process μ may have different variances $\parallel$ and $\perp$ to the longitudinal axis of a missile and this will be represented by the principal axes of R_μ being state dependent). The form of the CFP equation given above must be modified slightly to accommodate these changes.

b. The results of the last section can be readily extended to any number of sensor stations, in which case (3.15) can be replaced by

$$P_{1,\ldots,n}(x,t|y_1^t, \ldots, y_n^t) = a(t) \prod_{i=1}^{n} P(x,t|y_i^t)\, P(x,t)^{-n+1} \quad , \qquad (4.1)$$

where a(t) is a normalization factor.

c. The results of the last section were derived on the assumption of continuous conditioning (i.e., each set y_i^t corresponds to a set of measurements on a continuous set of times). The results can be easily reformulated for a discrete set of measurements. H_i needs only to be rewritten in the form

$$H_i = \sum_{k=1}^{p} \left(y_{ik} - g_{ik}(x)\right)^T C_{\nu ik}^{-1} \left(y_{ik} - g_{ik}(x)\right) \delta(t-t_k) \quad , \qquad (4.2)$$

using the set of measurement models

$$y_{ik} = g_{ik}(x) + \nu_{ik} \; ; \; i = 1,\ldots,n; \; k = 1,\ldots,p \quad , \tag{4.3}$$

where the ν_{ik} are Gaussian random vectors with the properties

$$E\nu_{ik} = 0, \; E\nu_{ik}\nu_{i'k'}^{T} = \delta_{ii'}\delta_{kk'}C_{\nu ik} \quad , \tag{4.4}$$

and where $x(t)$ and ν_{ik} are statistically independent when $t < t_k$. The extension to the mixed case involving a combination of discrete and continuous sets of measurements is straightforward. The extension to the asynchronous case, in which the sets of measurement times at different stations are in general different, is also straightforward.

d. When all the PDs are Gaussian, the conditioned means and covariances need to be transmitted to the central processor only when fusions are to be accomplished. If the types of measurements and their times of occurrence are known a priori, the covariances will be independent of the results of measurements and thus can be computed beforehand at a central station.

e. Parameterized acceleration models for maneuvering can be incorporated in the state vector x and then the general formalism discussed in the previous sections can be used. A flat-earth model of object motion in space could use a model of the form

$$
\begin{aligned}
\frac{d}{dt}\,\vec{r}(t) &= \vec{v}(t) \quad , \\
\frac{d}{dt}\,\vec{v}(t) &= -\vec{e}_z g + \vec{\alpha}(t) \quad , \\
\frac{d}{dt}\,\vec{\alpha}(t) &= \vec{\beta}(t) \quad , \\
\frac{d}{dt}\,\vec{\beta}(t) &= \vec{\gamma}(t) \quad , \\
\frac{d}{dt}\,\vec{\gamma}(t) &= 0 \quad ,
\end{aligned}
\tag{4.5}
$$

where $\vec{r}(t)$ is the position of the center of gravity of the object; $\vec{v}(t)$ is the corresponding velocity; $\vec{\alpha}(t)$ is the excess acceleration relative to the acceleration due to gravity, i.e., $-\vec{e}_z g$; $\vec{\beta}(t)$ is the rate of change of excess acceleration (sometimes known as "jerk"); and finally $\vec{\gamma}(t)$ is the rate of change of jerk or the acceleration of excess acceleration. The general solution of the above system can be written in the form

$$
\begin{aligned}
\vec{\gamma}(t) &= \vec{C}_4 \quad , \\
\vec{\beta}(t) &= \vec{C}_4 t + \vec{C}_3 \quad , \\
\vec{\alpha}(t) &= \frac{1}{2}\,\vec{C}_4 t^2 + \vec{C}_3 t + \vec{C}_2 \quad , \\
\vec{v}(t) &= \frac{1}{6}\,\vec{C}_4 t^3 + \frac{1}{2}\,\vec{C}_3 t^2 + (\vec{C}_2 - \vec{e}_z g)\, t + \vec{C}_1 \quad , \\
\vec{r}(t) &= \frac{1}{24}\,\vec{C}_4 t^4 + \frac{1}{6}\,\vec{C}_3 t^3 + \frac{1}{2}\,(\vec{C}_2 - \vec{e}_z g)\, t^2 \\
&\quad + \vec{C}_1 t + \vec{C}_o \quad ,
\end{aligned}
\tag{4.6}
$$

in which $\vec{C}_o, \ldots, \vec{C}_4$ are constant vectors. We will define the non-maneuvering (i.e., pure ballistic) solution to be given by $\vec{C}_2 = \vec{C}_3 = \vec{C}_4 = 0$ in which case we obtain the familiar result

$$\vec{\alpha}(t) = \vec{\beta}(t) = \vec{\gamma}(t) = 0 \quad ,$$

$$\vec{v}(t) = -\vec{e}_z g t + \vec{C}_1 \quad , \tag{4.7}$$

$$\vec{r}(t) = -\frac{1}{2}\vec{e}_z g t^2 + \vec{C}_1 t + \vec{C}_0 \quad ,$$

The deviation from this motion is given by the incremental solution

$$\vec{\gamma}(t) = \vec{C}_4 \quad ,$$

$$\vec{\beta}(t) = \vec{C}_4 t + \vec{C}_3 \quad ,$$

$$\vec{\alpha}(t) = \frac{1}{2}\vec{C}_4 t^2 + \vec{C}_3 t + \vec{C}_2 \quad ,$$

$$\delta\vec{v}(t) = \frac{1}{6}\vec{C}_4 t^3 + \frac{1}{2}\vec{C}_3 t^2 + \vec{C}_2 t \quad ,$$

$$\delta r(t) = \frac{1}{24}\vec{C}_4 t^4 + \frac{1}{6}\vec{C}_3 t^3 + \frac{1}{2}\vec{C}_2 t^2 \quad . \tag{4.8}$$

Clearly, this solution can be regarded as a parameterized maneuvering model with the nine constants $\vec{C}_2$, $\vec{C}_3$, and $\vec{C}_4$ regarded as the parameters.

This model can be regarded as a special case of the dynamical model discussed in Section 2 with (4.5) replacing (2.1). Here, all of the randomness is in the initial conditions as is the case in t! general dynamical model given by Eq. (2.1).

f. When the maneuvering model involves a random process, e.g., (2.1) is replaced by

$$\frac{dx(t)}{dt} = f(x(t),t) + \mu(t) \quad , \tag{4.9}$$

where $\mu(t)$ is a Gaussian random process with the properties

$$E\mu(t) = 0,\ E\mu(t)\mu(t')^{T} = R_{\mu}\delta(t-t')\ , \tag{4.10}$$

the arguments in the last section break down, due to the appearance of a diffusive term $-\frac{1}{2}\frac{\partial^{T}}{\partial x} R_{\mu} \frac{\partial}{\partial x} P_{12}$ in the CFP equation (2.10) with similar modifications in later versions of the CFP equation. The arguments leading to the fusion equations (3.13) and (3.15) depend upon the operators involving x being at most first order in $\frac{\partial}{\partial x}$.

g. In the above case, it is sufficient to transmit the $H_i = H_i(y(t),x,t)$ continuously (or intermittently in the case of discrete sets of measurements) to a central station. However, an acceptable approximation in the continuous case is to transmit time-integrated versions of H_i intermittently if the intervening time intervals are not too long. A currently unsolved problem is related to the appropriateness of transmitting the $P_i(x,t) \equiv P(x,t|y_i^t)$ at a continuous or discrete set of times.

h. The extension of the above analysis must be considered from the case of one moving object to the more interesting case of many moving objects. To deal with this question we must consider the formulation of an appropriate representation of the many-object state. For the sake of simplicity, the objects are assumed to be indistinguishable in that any system of labelling is artificial and that any functions (in particular, the PDs) of the state must be invariant to permutation of labels. Let us define the many-object state x by the set $x = \{(N,z_1,\ldots,z_N)|N = 0,1,\ldots,\infty\}$, where z_q is now the state of object q and N is the number of objects in a particular situation. When $N = 0$ we will set $(N,z_1,\ldots,z_N) = (0)$. If only one object is present then $x = z_1$ and $N = 1$, which is the case treated in the last two sections. The PD of x is now given by the set $\{P_N(z_1,\ldots,z_N,t)|N = 0,1,\ldots,\infty\}$ in which $P_N(z_1,\ldots,z_N,t) = P_o$, a time-dependent constant, when $N = 0$. For every value of N the corresponding CFP equation can be easily formulated, but with the normalization involving all values of N. From a formal abstract

point of view the arguments will proceed in the same way as the single object case. The assumption that $P_N(z_1,...,z_N,t)$ is invariant to the permutation of the subscripts of $z_1,...,z_N$ (i.e., it is a symmetric function of $z_1,...,z_N$) greatly facilitates the fusion process by eliminating such processes as data association.

i. An important point is that parametric maneuvering models are not as restrictive as they might appear at first sight. From the differential equation $\dot{x}(t) = f(x(t),t)$ a set of deterministic trajectories can be deduced. In the absence of conditioning the most probable trajectory will be a member of the above set. However, with conditioning present, the most probable trajectory is almost never a member of this set; i.e., it is almost never a solution of the above differential equation. In the dynamical model given by Eq. (2.1) to (2.4), the most probable trajectory is almost never a power series in t. This somewhat anti-intuitive situation is understandable when one realizes that the process of conditioning continually revises the probabilities of the different trajectories in the above deterministic set so that (speaking loosely) one short segment of one trajectory is most probable in one time interval while another segment of another trajectory is most probable in a different interval.

j. Another viewpoint on this question is provided by the linear Gaussian case. Here, the evolution of the CPD is given by the Kalman-Bucy[9] equations since this function remains Gaussian if it is initially and thus it is fully represented by the conditional mean m(t) and covariance C(t). Assuming that the dynamical and measurement models are given by the equations

$$\dot{x}(t) = F(t)\, x(t) + \mu(t) \,, \tag{4.11}$$

$$y(t) = G(t)\, x(t) + \nu(t) \,, \tag{4.12}$$

$$E\mu(t) = 0;\ E\mu(t)\mu(t')^T = R_\mu(t)\ \delta(t-t')\ , \tag{4.13}$$

and that $\nu(t)$ is still defined by Eqs. (2.3) and (2.4), then the CPF equation reduces to the Kalman-Bucy equations

$$\dot{m} = Fm + K(y - Gm)\ , \tag{4.14}$$

$$\dot{C} = FC + CF^T - CG^T R_\nu^{-1} GC + R_\mu\ , \tag{4.15}$$

$$K = CG^T R_\nu^{-1}\ , \tag{4.16}$$

where $K = K(t)$ is the so called Kalman gain. Under the earlier assumption that the dynamical model is deterministic, i.e., $\mu = 0$, the term R_μ in Eq. (4.15) would vanish. Here, the conditions under which R_μ can be neglected when it does not vanish must be determined. R_μ has a rather indirect effect on the evolution of C, which in turn has an effect, through the Kalman gain K, on the evolution of m by determining the response to the quantity y - Gm. Going back to the evolution of C, if $R_\mu << CG^T R_\nu^{-1} GC$ in some sense, the effect of R_μ is very small. However, in the course of time the matrix on the r.h. side of the above inequality will decline (according to some suitable measure of magnitude) until it becomes comparable to R_μ on the left-hand side, after which R_μ is not negligible. Unless the dynamical model contains damping (which in an exo-atmospheric environment could occur only in the parametric maneuvering module), the effect of R_μ should become increasingly dominant. Thus if C(0) is sufficiently large to satisfy the above inequality, there is a certain interval (starting with t = 0) of time in which the effect of R_μ is negligible and the length of this interval increases as C(0) increases.

k. The treatment of the fusion problem has so far ignored time delays associated with finite electronic processing speeds, finite propagation speed of the signals transmitted from one station to another, and finally the finite propagation speeds involved in the physical interactions between the sensors and the objects of interest. If the third cause of time delay did not exist, then we would have no problem since the delays associated with the electronic processing and communication are presumably knowable locally with sufficient accuracy. However, delays arising from the third cause are not readily inferrable at a single sensor station based upon the local sensor outputs. This inference process is still more difficult in the case of passive measurements. Here, such delays can perhaps be inferred from the probability density in state space conditioned upon all of the available raw data. A certain amount of circularity is involved in this inference process since the local processing at each sensor station must also deal with time-delay questions that can only be answered globally, if at all. Clearly, substantial further work must be done on this problem.

l. In Section 3 we proved that $P_{12}(x,t) = a(t)P_1(x,t)P_2(x,t)P_o(x,t)$, for two sensor stations. Here the full state vector x is involved in each probability density. The question arises, can x be replaced by lower dimensional vectors in $P_1(x,t)$ and $P_2(x,t)$ associated with sensor stations 1 and 2, respectively? In some simple cases it can be. For example, in the case of a ballistic object in a vacuum above a hypothetical flat earth this can easily be done by two observers viewing the motion of the object along lines of sight at right angles to each other. A more general form of this dimensionality reduction procedure has been recently derived.[10]

5.0 DISCUSSION

The results derived in the previous sections are on an abstract level where proof of principle takes precedence over problems of implementation. We must now turn to consider the practical problems entailed in implementation. First of all, clearly some kind of approximate representations of the probability densities (PDs) must be devised so that approximate versions of P_1, P_2 can be transmitted from the sensor stations 1 and 2 to a fusion station where the total CPD P_{12} is computed using an approximate version of the formula $P_{12} = a\, P_1 P_2 P_0^{-1}$ where a is a normalization factor. If all of the CPDs are Gaussian the problem is straightforward. In the non-Gaussian case (due to nonlinearities in dynamical or measurement models, or to the presence of several objects) more than means and variances must be used to represent the CPDs (e.g., use Gram-Charlier series, Gaussian sum approximations). Returning to the question of many objects, it may be more appropriate to consider various orders of conditional mean densities in single-object state space and an associated hierarchy of equations giving their time evolution. Various approximation methodologies must be devised for reducing this formulation to computationally tractable form.

In any case, the idea that the transmission of CPDs from sensor stations to a fusion station entails less load on the communication system than the transmission of the raw data depends, of course, on two factors: (1) how often the CPDs are transmitted and (2) what kinds of approximate representations of the CPDs are used. We have assumed that these two factors can be handled such that it is distinctly advantageous to transmit approximate reprsentations of CPDs.

Another point to consider is that in the transmission of CPDs no supplementary information is required concerning what measurements were made and when they were

made. On the other hand in the transmission of raw data all supplementary information is essential.

When noise in the dynamical model cannot be neglected even with an extended state space, an intermediate approach is to transmit H_i, $i = 1,2$, suitably integrated over appropriate intervals of time. The reader is reminded that H_i is defined by

$$H_i = H_i(x,t) = (y_i(t) - g_i(x,t))^T R_\nu^{-1} (y_i(t) - g_i(x,t)) \quad . \tag{5.1}$$

It can be replaced by a reduced version with state-independent terms ignored. Approximate representations of H_i must also be developed.

6.0 PROBLEMS REQUIRING FURTHER RESEARCH

Many additional problems require considerable further research effort, at both the abstract and the implementation levels. Some of these are:

a. Inclusion of noise in dynamical model.

b. Handling of general sequences of continuous and discrete sets of measurements.

c. Treatment of unknown time delays between object and sensors.

d. Full development of the many-object case.

e. Redundancy in the storage of fragments of the state of knowledge.

Acknowledgments

The author thanks his colleagues K.A. Marsh and A.R.K. Sastry for several penetrating comments.

REFERENCES

(1) R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Trans. ASME, Series D, Vol. 83 (1960).

(2) S. Mori, C.Y. Chong, E. Tse, and R.P. Wishner, "Multitarget Multisensor Tracking Problems - Part 1: A General Solution and a Unified View on Bayesian Approaches," AI and DS Technical Memo, TR-1048-01 (1984).

(3) C.Y. Chong and S. Mori, "Fusion Algorithms for Hierarchical Multitarget Tracking," Proc. 7th MIT/ONR Workshop on C^3 Systems, pp. 179-184 (1984).

(4) Y. Bar-Shalom, "Tracking Methods in a Multitarget Environment," IEEE Trans. Automat. Contr., Vol. AC-23, pp. 618-626 (1978).

(5) D.B. Reid, "An Algorithm for Tracking Multiple Targets," IEEE Trans. Automat. Contr., Vol. AC-24, No. 6, pp. 843-854 (1979).

(6) J.L. Doob, "Stochastic Processes," Wiley, New York (1967).

(7) H.J. Kushner, On the Differential Equations Satisfied by Conditional Probability Densities of Markov Processes, with Applications, J. SIAM on Control, Series A, Vol. 2, pp. 106-119 (1964).

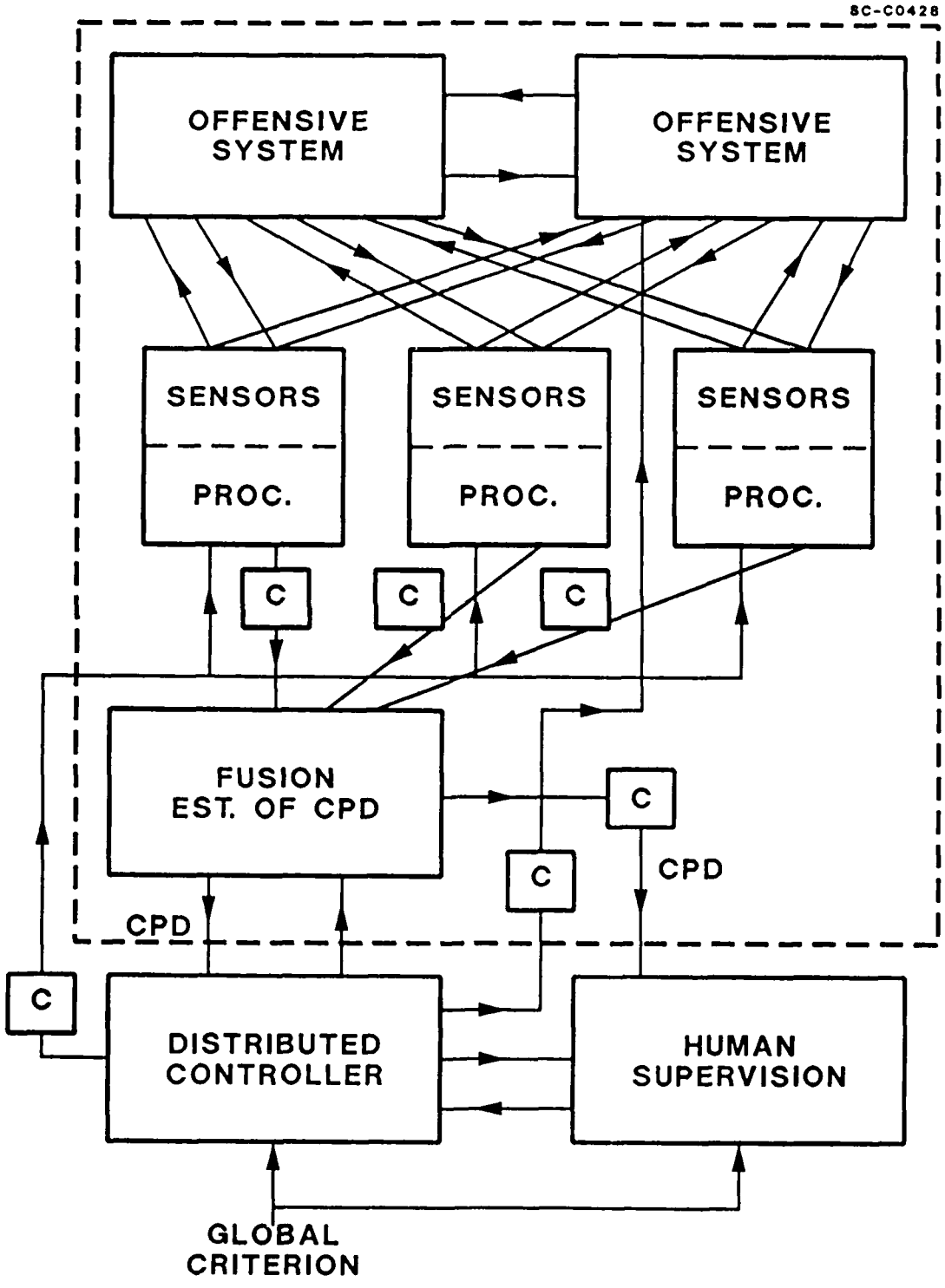
(8) R.S. Bucy, Nonlinear Filtering Theory, IEEE Trans. on Autom. Control, Vol. AC-10, p. 198 (1965).

(9) R.E. Kalman and R.S. Bucy, 1961, New Results in Linear Filtering and Prediction Theory, Trans. ASME, Series D, Vol. 83, pp. 95-108.

(10) J.M. Richardson, to be published.

Figure Captions:

Fig. 1 Schematic representation of distributed decision making (details in text).


SC-C0428
OFFENSIVE SYSTEM
OFFENSIVE SYSTEM
SENSORS
PROC.
SENSORS
PROC.
SENSORS
PROC.
C
C
C
FUSION EST. OF CPD
C
C
CPD
CPD
C
DISTRIBUTED CONTROLLER
HUMAN SUPERVISION
GLOBAL CRITERION

APPENDIX III

****
ISL Working Paper — ISL-WP-89-3, November 22, 1989
Comments Welcome
****

APPENDIX III

Dynamic Organizational Rationality:

Applying A Framework for Thinking about and Measuring Group Effectiveness to Command and Control

by

Michael R. Fehling and Michael J. Lippitz†
Intelligent Systems Laboratory
Dept. of Engineering-Economic Systems
Stanford University
Stanford, CA 94305-4025

ABSTRACT

Beginning with a set of primitive notions of group structure and evolution, we define concepts of rational agency, organizational intelligence, and autonomy. These definitions, linked to measures of performance that follow naturally from models of perfect rationality, support the development of a normative functional theory of organizations as problem solving systems, where "problem solving" is understood to be the *process* of reasoning about and taking rational action to achieve some objective. The structural point of view implied by these new definitions is what we are calling The Organization Metaphor. The functional implications of The Organization Metaphor are developed via a notion we call "agreements". Taken together, these conceptions represent a theory of Dynamic Organizational Rationality with which we confront some important and long-standing issues in conventional Organization Theory. We then describe Schemer, a computational model of resource-constrained problem-solving agents, and discuss how Schemer operationalizes and provides a formalizable framework within which our notions of Dynamic Organizational Rationality may be explored. We conclude with a discussion of our theory and its technical implications for modeling and analyzing intelligent activity in distributed systems, particularly military command-and-control systems.

---

†Most of this work was performed while the authors were members of the research staff of the Rockwell International Science Center, Palo Alto Laboratory for Intelligent Systems. At Rockwell this research was supported under contract no. N00014-87-C-0703 from the Office of Naval Research. Since September 1, 1989 Fehling has been a member of the faculty of Stanford University, Dept. of Engineering-Economic Systems. Lippitz is now pursuing full-time graduate studies in that academic department.

Table of Contents

1. Introduction.

It has been said that understanding organizations will be as critical to modern society as understanding agriculture was to our predecessors.[1] Yet a clear, unified notion of group problem-solving remains elusive, and the study of organizations as a formal discipline is widely considered to be in a pre-paradigmatic state (cf., Pfeffer, 1985). Motivated in part by the extant literature's rather bewildering array of selective viewpoints of, and technical approaches to, this topic, we present in this paper some early thoughts on a new theoretical view of organizations and their activities that exploits concepts and methods we have developed in previous research on the computational modeling, analysis, and implementation of complex, distributed problem-solving systems (Abrams et al., 1984; Fehling & Erman, 1983). Our approach and the perspective that it embodies are interdisciplinary, combining theoretical commitments from the field of Artificial Intelligence (AI) toward modeling intelligent agents as computational processes with normative commitments of decision analysis and behavioral decision theory (Howard, 1968; Raiffa, 1968; von Winterfeldt & Edwards, 1986).[2]

This initial discussion explores some foundational issues underlying alternative viewpoints and differential emphases in the study of organizations. Based upon these foundational considerations, we present a more unified theoretical framework with a particular focus on the factors that underlie an organization's ability to act coherently as an effective problem-solving "agent" in its task environment. Our consideration of what constitutes organizational effectiveness follows naturally from this development.

This latter task is theoretically deep and challenging. A sound, general basis for measuring group effectiveness derives ultimately from appropriate prescriptions that determine (at least ideally) what constitutes correct, i.e., *rational*, action. Recognizing the difficulties entailed herein, we nevertheless seek to understand in what sense (if at all) an organization can be said to act rationally and reflect the rationality of its constituent members. To our knowledge, this objective has not been faced squarely in existing theory. In order to provide a proper basis from which specific concepts and measures of effectiveness may be developed, we begin our inquiry with some very fundamental questions about an organization's purpose and function:

1. Why, or under what conditions, do organizations exist?
2. How are organizations as entities related to the individuals that constitute them?
3. What form of intelligence or rationality, if any, can one attribute to organizations?

Having laid a theoretical foundation, we then describe a computational framework for formal modeling of organizational structure and dynamics.[3] In this framework, organizations and their constituent sub-organizations and individuals are conceived as distributed, interconnected systems of information processing and action-taking elements.

---

[1]From remarks by Richard Scott, Director, Stanford Center for Organizational Research, in an inverview published in The Stanford Daily, Nov. 14, 1988.

[2]We shall attempt to distiunguish our decision-analytic view from that of team decision theory (e.g., Kim & Roush, 1987).

[3]One very important long-term goal of this research is to demonstrate the value of using a computational approach to modeling for elucidating and analyzing the important features of organizational dynamics.

(This description includes, but is not restricted to, organizational elements that one thinks of as intelligent, problem-solvers.) Also embedded within our architecture are the organization's methods that determine its reasoning and action-taking abilities, its means of encoding and storing information, its "control" policies for managing its resources and exploiting available external resources in its environment, and finally a precisely defined representation of how an organization interacts (e.g., communicates) with its environment.

In previous research, and during this project, we have formulated and refined a very general computational architecture called Schemer (Fehling et al., 1989b) that meets these requirements.[4] Schemer constructs are sufficiently rich to support (a) precise formulation and analysis of dynamically changing communication relationships both within and among organizational elements, (b) procedural description of the task-performance and problem-solving capabilities of individual elements and groups of elements, and (c) a uniform articulation of the policies by which an organization enforces selective control over its actions in response to external and internal constraints. Schemer can therefore serve as a computational framework within which to model a wide variety of complex, distributed systems that include cooperative or competitive "intelligent agents" as subsystems (e.g., Abrams et al., 1984).

This commitment to computational modeling, which is perhaps the most salient feature of our approach to organizations, addresses three very important concerns that are both methodological and substantive. Schemer provides a modeling framework[5] that

- affords precision in describing organizational structure at a wide range of levels of detail,
- is well-suited to describing the wide variety of *processes* carried out within and among organizations, in an organization's environment, and in organization-environment interactions; and
- provides sufficiently expressive basis for modeling the *intentional*[6] characteristics of the *processes* underlying actions and deliberative in a problem-solving agent, be it individual or group.

Schemer thus provides two important senses of expressive generality —(1) the ability to model arbitrary types of organizational elements, ranging from human decision-makers to physical components such as communication or computational elements; and (2) a framework within which all aspects of an organization can be modeled, including systems and policies for inter- and intra-organizational communication, management (or command) and control, beliefs and hypotheses (i.e., intelligence), and actions by which the organization can operate on its environment. Schemer will, we hope, improve the clarity

---

[4]The word architecture as we are using it is ambigous. It could refer to an abstract description of a class of designs for computing *hardware* or, alternatively, a type of software design, or some combination. In other papers we tyically describe Schemer as a software architecture (e.g., Fehling & Breese, 1988). However, this distinction may be dropped in the present discussion. Schemer is the formulation of a "virtual machine" (Fehling et al., 1989b), that is, a basic set of computational features and processes in terms of which all more complex elements are defined and within which these elements are embedded and managed. This virtual machine description may be thought of as physical hardware or as a software system whose features are implemented in some other hardware environment.

[5]I.e., a virtual machine architecture with a language in terms of which to formulating descriptions of a modeled system's structure and processes.

[6]I.e., described in terms of willfull or purposeful commitments to *rational* action. The inclusion of rationality is, of course, a significant extension to discussions of intention typically found in the philosophical literature (e.g., Anscombe, 1963; see however, Bratman, 1987).

and precision of the theoretical concepts and operational terms that we use in our investigation.

2. Historical Motivation.

Our investigation of organizational issues and, in particular, our quest for a general methodology for modeling and analysis, was initiated by our participation in a research initiative entitled "Distributed Tactical Decision-Making" (DTDM). This initiative, sponsored by the Office of Naval Research, was concerned with bringing coherence to ad hoc studies of organizational dynamics by facilitating collaboration among researchers developing various methods for evaluating alternate command structures, communication protocols, and other command and control (C^2) subproblems. Research on such organizational problems such as C^2 is limited by the piecemeal nature of methods and solutions being developed, each of which focuses on a different aspect of the C^2 problem and each of which tends to be developed under idiosyncratic assumptions and in a specialized language. We were asked to take a broad view of research in the field and lay the groundwork for a unified framework that would help reveal relationships among the subproblems and suggest ways to solve large-scale organizational design problems in a principled manner.

An example of such a large scale problem in the DTDM realm would be the evaluation of the trade-off between the efficiency of a rigid hierarchical organization in completing well-defined tasks versus the adaptability and throughput advantages of an organization in which decision making authority is distributed (cf., Drenick, 1986). Discussions at the 1988 DTDM coordinating meeting regarding the advantages and disadvantages of the current command hierarchy in which different functional officers report to a Combined Warfare Commander concerned precisely this trade-off.

As we have already noted, our technical background and interest in the design of generic computational models strongly influences our thinking and distinguishes us from other researchers who have contemplated C^2 issues. Specifically, it predisposes us to view organizations as problem solving systems, where "problem solving" is understood to be a resource-constrained *process* of reasoning about and taking action towards objectives. This perspective highights the nature of "control" as a trade-off between the *functional* advantages of autonomy versus those of interdependence. Because complete autonomy is essentially a foundational assumption of conventional analytic models that emphasize structural optimization, we found it difficult, if not impossible, to express this trade-off using their mathematical formalisms. However, such theories provide a formal basis for evaluation of group performance and suggest key problems groups must overcome in making decisions—e.g., information disparities, value conflicts, etc. We thus draw heavily from them in order to give normative force to our theory of the process by which, in the words of Jay Lawson, "a team of experts becomes an expert team."

The formulation of some well-founded notion of organizational competence and our commitment to development of generic approach to modeling complement each other and are clearly interrelated. However, our progress on these two objectives differed markedly. We began to make immediate and rapid progress in developing a generic modeling framework based on the Schemer computational architecture. While this framework is as

yet untested for such a general purpose as this one,[7] it served immediately as a convenient basis for refining our ideas. In fact, as we began using Schemer's constructs as a basis for describing C^2 organization and C^2 processes, we developed increasing confidence in its suitability as a generic framework for modeling organizational issues. Indeed, as we prgressed with our research, we found it necessary to make only minor refinements to Schemer when modeling specific examples of C^2 systems and scenarios exemplifying C^2 processes.[8]

On the other hand, it became clear to us at an early stage that our second objective posed far deeper challenges. Although our computational framework provides a very robust, expressive basis for describing organizational elements and the processes by which they reason, make judgements, choose, and take action in the task environment, it offered only limited guidance on *standards* in terms of which these processes and their results could be assessed.[9]

We turned to the literature on organizations, organizational processes, and organizational behavior (e.g., Pfeffer, 1985). There we found a vast and diverse array of analyses and perspectives, almost as numerous as the distinct authors who contribute to that literature. We shall not review that literature here. We merely note that our review produced no unifying (nor even a single dominating) theory, perspective, or model that we could exploit in our current work. In fact, we found ourselves facing a conundrum. On the one hand, intelligent, individual humans obviously are motivated to organize themselves into effective groups and routinely do so, at least under some conditions. On the other hand, no theory has clearly offered some precise notion of how or under what conditions the organizing process takes place. Nor have they provided the basis by which members of a group (or others) may determine group effectiveness and how the activities of the group's members may be understood and measured. But to provide a generic, normative basis for evaluating a group's *as an entity* in its process of judgement, reasoning, and actions, we require a suitable concept of group effectiveness akin to the view of individual rationality offered by the great theorists of decision making (e.g., Ramsey, 1965; Savage, 1954; Goode, 1983). In other words, we seek a prescriptive basis for group rationality.

We were thus led to engage in the conceptual and philosophical examination described in the next section. (The need for this level of re-examination no doubt explains why it is that no coherent concept of organizational rationality is yet available.) We were convinced that group rationality is closely related to the rationality of the individuals that comprise the group. However, we want to avoid inventing a separate "social force" that magically provides a new set of principles that govern group behavior. We sought instead a way of characterizing group rationality as stemming from the rational actions of the group's members. Thus, a group must be seen neither as a mere aggregation of individuals taking action, nor as some radically new kind kind of intelligent being whose performance is

---

[7]We had planned to use the results of IR&D project 837 at the Rockwell International Science Center to implement Schemer and test its suitability as a framework for modeling C^2 commication techniques and other specific C^2 issues. Unfortunately, Science Center management decisions forced abandoning that goal.

[8]This corroborates the claims made by Abrams et al. (****) in an early investigation of Schemer's usefulness for modeling the organizational dynamics of C^2 systems.

[9]Schemer does offer at least some guidance in criteria for effectiveness, albeit very weak ones. As is common in AI problem-solving models Schemer enforces a "local" consistency constraint on the information being simultaneously used to carry out some task. In particular, if potential inconsistency is detected then the most important task immediately becomes one of elimating that inconsistency. However, the problem-solver is allowed to retain and separately use sets of facts that are mutually inconsistent. Thus, a "global" consistency constraint on the problem-solver's information is not enforced.

based on different standards than the problem-solving beings of which it is constituted. In order to accomplish this task, we re-examined some very basic issues regarding individual and multi-agent action. The alternative would have been to abandon our objectives and perhaps aim for simpler (and more *ad hoc*) descriptive methods. Thus we pressed on.

"Why, and under what conditions, do organizations exist?" was the seminal question for our research. Before approaching the "why" part of the question, it was necessary to be clear about what constitutes the "existence" of an organization. In other words, what is it about a group that leads one to recognize it as a unified entity that is relevantly different than the simple sum of its parts? Going one step deeper, one must ask what it means to be "unified"? We answer this question with a primitive commitment: For an entity to be considered unified, it must behave in a way that is consistent with its beliefs, values and goals. Beliefs, values and goals—also primitive notions—are derived, through a process of reflection, from examining the coupling between the entity and its environment. The concept of reflective understanding is thus central to our development.

For an individual problem solver, the axioms of rationality provide a basis for judging how well beliefs, values and goals are reflected in behavior. We thus proposed a second foundational question: "What form of intelligence or rationality, if any, can one attribute to organizations?" This question led us to examine the basic problems organizations must overcome to survive and provided a basis for measuring group effectiveness analogous to that which would be applied to individuals.

A deeper consideration of organizational goals and values led us to seek out the basic motivations toward group formation and the criteria which underwrite and (ideally) propel its intentional activity. Clearly an organization's "mental attitudes" (i.e., its goals, values, and the beliefs to which they are related though action) must be related in some way to the similar mental attitudes embodied in the group's members. In this way, our investigation led naturally to a third foundational question: "How are organizations as entities are related to the individuals that constitute them?" Our answer to this question represents the heart of "the Organization Metaphor" discussed in the next section. We express this metaphor in terms of primitive structural notions of "agency" and "autonomy."

We then turned our attention to functional mechanisms that, when further refined, can transform the Organization Metaphor into a proper theory of rational group action. This new focus helped us sharpen our approach to the third foundational question: "*How* do organizations act rationally?" Section 4 explicates the mechanisms agents use to solve their organizational problems. We introduce a basic organizing mechanism suggestively called "agreements" and sketch a theory of their operation.

Taken together, the Organization Metaphor and the notion of agreements express our conception of Dynamic Organizational Rationality. The concepts of agreements, reflective understanding, limited rationality[10], agency, and autonomy ground our basic theoretical framework for studying the nature and effectiveness of organizations, organizational learning and evolution.[11] In Section 5 we describe Schemer, our computational framework for modeling group structure and processes. Finally, in Section 6, we briefly

---

[10]Briefly, *limited* rationality refers to a standard of rational action that takes into account the agent's ability adapt to contextually imposed constraints on reasoning and action. These constraints come from the limitations of resources such as time, information, cognitive processing capacity, etc. and bound the agent's abilities. In short, the agent must choose the best action where the costs of deliberative choice are considered as well as the costs and benefits of the chosen action (cf., Goode, 1983).

[11]Further consideration of resource contraints could provide a formal restatement of sociological theories of power.

discuss how we expecting to bring together the two major aspects of this work, operationalizing our evolving theory of dynamic organizational rationality within Schemer's computational modeling framework.

3. The Organizational Metaphor.

Before we consider organizations specifically as social systems (i.e., systems that coordinate the activities of multiple, independent intelligences), we begin by examining the concept of organization as a manifestation of intelligence and rational action by autonomous problem-solving agents.

Groups and individuals, viewed as intelligent problem-solving entities or "agents," possess some degree of *autonomy* in their relationship to other agents and to their environment. Autonomy, as we conceive it, is a direct product of the ability of an agent to reflectively adapt to its environment. Things that cannot (or do not) reflect are fully determined by their a priori capacities and the conditions in their environment. If we can determine these capacities, then, in principle, we can determine the behavior of the system under all conditions. More importantly, if we can determine these capacities, then we can control these systems by managing environmental conditions. In this sense, things that cannot reflect are deterministic.

To be autonomous, therefore, is to be reflective.[12] Things that can reflect on their relationship to their environment (and use the results of reflection to change that relationship) possess autonomy. The capacity for reflection thus sets autonomous agents apart from other entities in the world. The quality of an agent's reflective apparatus and the resources it has available for reflection are measures of its autonomy. Autonomy is, thus, *not* an "all-or-none" feature of an agent; it is an inherently variable feature describing the extent to which the agent is free of control by external forces.

We view *intelligence* as a measure of the ability of an agent to *understand* its environment (and itself). Understanding is a birthing process. That is, understanding is a measure of an agent's capacity to adapt to situational idiosyncrasies, to recognize regularities, and to learn how to exploit them. Adaptation and learning are measured (by the agent through reflection or by another agent through observation) as increases in an agent's capability for taking actions, typically actions that meet some part of the agent's goals. This connection of understanding to action leads one to postulate standards of *rationality* that are inextricably interwoven with a concept of intelligence and the processes of understanding (cf., Ramsey, 1965). A normative concept of rationality represents an explicit standard for judging how well behavior is consistent with an agent's evolving beliefs, desires, and intentions (cf., Anscombe, 1963).

In sum, intelligence as we view it, is an indirect measure of an agent's ability to improve the effectiveness of its actions with respect to its objectives. At the very lowest levels, it is straightforward to apply normative standards directly in the measurement of action. It is less direct to recognize and apply such a standard to the more global measure of an agent's intelligence. Nevertheless, some such standard does apply. To measure the intelligence of an agent (individual or group) is to assess the effectiveness of its actions, most importantly its efforts to improve the effectiveness of its intent-directed actions. So, the capability for understanding, of which intelligence is the general measure, depends as much on reflection

[12]Entities that are deterministic but prohibatively complex are sometimes better modeled as autonomous because of our own cognitive limitations.

as does autonomy. It is in the processes of reacting, deliberating, and choosing that intelligence may be observed. Understanding and intelligence are must therefore be understood as *functional* processes of an agent, not simply a compendium of previously learned skills.

Understanding is also an organizing process. As a reflective agent comes to develop increased capabilities, that agent manifests enhanced *knowledge*. Knowledge is a collection of beliefs that the agent exploits using reflective deliberation to further improve its effectiveness in acting. Knowledge is composed of many elements and, equally important, stipulated relationships among these elements. These relationships serve to organize knowledge into a system that helps to determine the meaning of any element of that system. For example, we are quite used to thinking of the categorical and taxonomic relationships among elements of a system of descriptive beliefs. Interestingly, research in AI and cognitive science has been especially helpful in elucidating how this sense of organization holds even among an agent's dispositions and skills, which are organized according to such indices as the conditions of their use, constraints upon their effectiveness, and their co-occurence and interaction when enacted as processes.

What motivates an intelligent agent to reflect on its experience in the first place, and what criteria might it apply in developing organization? We take a normative perspective and assume that an agent's drive to obtain mastery over its environment is determined by deeper needs the agent must meet in in order to survive. Through the process of reflection, agents then develop a host of derived needs and preferences such as value, preference and utility that are posited in the decision-theory literature. Reflective agents, in effect, pull themselves up by our own bootstraps.

Our remarks have led us to an interesting position. We have now seen that an intelligent agent applies reflective understanding to improve the effectiveness of its actions through adaptation and learning. This results in an organization within the agent itself. This is not merely the taxonomic organization of an agent's beliefs. It includes the organization of an agent's capabilities for acting. These dispositions form a structured collection of specialized skills, each at the ready for carrying out its own unique role. We seem to have found that an agent, viewed as an individual, comes upon closer inspection to be seen as really an organization of interacting specialists (Erman et al., 1980; Fehling et al., 1989a).[13]

We thus stake a claim in opposition to individualistic perspectives, resolving the micro-macro split by dissolving it. We hold that "organization" is ubiquitous. To think or to act with purpose is to organize, and any entity that can act with purpose may be thought of as an organization. We use the term "agent" to denote such entities, which can be viewed as individuals or groups. (These alternatives represent optional perspectives that one may choose in observing or analyzing the agent). One may analyze the organizations that result from reflection *structurally* by examining the relationships and groupings made among elements of experience as well as *functionally* by examining the process by which agents control their own reasoning and become more efficient at obtaining mastery. Taken together, these investigations will give substance to the sense in which agency as a *group* phenomenon can be thought of as normatively "rational."

---

[13]Some recent theorists (e.g., Baars, 1988) adopting this point of view go to some lengths to explain why it is that an "individual as society" retains such a strong sense of undecomposable self. Others, like Minsky (****) seem to be unconcerned by the apparent tension between the notion of self and organizational concepts of intelligence such as the "society of mind."

We have said that agents, be they individuals or groups, *create* organization through the process of understanding their experiences, their environment, and themselves. In turn, these agents *impose* that organization directly upon the environment by taking actions are based upon their understanding. Before understanding is manifest, the "river of experience" is inchoate and largely unstructured. An agent can, at best, react to this flow. Through understanding, an agent's relationship to experience can become interpretive, proactive and even directive. Understanding is, as Webster's defines it, "the power to make experience intelligible."

To say that agents create organization is, in effect, to claim that agents create other agents. This contention is clearly consistent with the systems approach to organizational modeling (Drenick, 1986) in which one seeks to discover isomorphic structures that describe each level in an organizational hierarchy. In a previous paper (Fehling & Lippitz, 1988), we mentioned our work on a recursively defined architecture for intelligent problem-solving systems, Schemer, that we are developing to serve as an isomorphic structure for agency. More deeply, Schemer hints at a theory of the evolution of organizational structure.

It should be noted that this point of view may not be reduced to ideas such as the "social-construction-of-reality." The fact that the environment is tangibly modified and structured through agents' actions based on their understanding provides the basis for communication and inter-subjective agreement among agents. An essential element of an agent's struggle to cope with the "river of experience" is coming to understand patterns in that experience that result from other agents' actions. There is thus no coherent basis for radically distinguishing "objective" from "subjective" realities. Any agent may be viewed either as an "individual," as a part of the environment for other agents, or as an "environment" for the specialized "agents" that constitute this agent's skills and knowledge. Patterns in the environment provide information merely as unstructured experiences which, when organized by an intelligent agent, become knowledge.

Multiple agents can exploit each others' capabilities by sharing information and coordinating their actions to achieve composite or common objectives. We will use the term "agreements" to denote protocols that separate agents may share and that govern the ways in which information is shared and the actions that such information evokes. Agreements cover the range of such protocols, from simple bilateral coordination to the institutionalized forms of understanding that provide the basis for large-scale social structure.

These definitional constructs — agents, agreements, and environment — are respectively the structural, functional, and contextual concepts that form the cornerstone of a new way of thinking about and modeling intelligent systems. We call this way of thinking "the Oganizational Metaphor." Beginning with the pristine notion that reflectively-generated organization is the basis of understanding, we posited the existence of fundamental needs (our normative perspective) to develop a unified conceptual notion of "agency" and "environment" in which agreements turn the cognitive organization of the agent into the social organization experienced as "environment."

These commitments represent the beginnings of "answers" to two of our foundational questions put forward at the beginning of this section: "Why, or under what conditions, do organizations exist?" and "How are organizations as entities related to the individuals that constitute them?" (We defer discussion of our third basic question to the next section. There, and again in section 6, we discuss the additional foundational and technical commitments that shape our answer to that question, "What form of intelligence or rationality, if any, can one attribute to organizations?") Organizations exist to enhance the effectiveness of action. That action springs from the perceived desires of the individuals

who recognize an inability to act on their own and who find a common basis for acting together. Agreements represent the basis formulated by these agents for coordinating their actions and their separately held information. If individuals were omnipotent they would never organize. If individuals were unadaptive they would be unable to organize. Organizations come to exist as an extension of the understanding (i.e., learning and adaptation) processes of the individuals that constitute them. Agreements are the constructive tools of this organizing process, i.e., they are the *functional implications* of agency.

Organizations and the individuals that constitute them are also related in terms of the individual and organizational levels of knowledge, desires, and determined goals. However, there is often a tension among these elements in that individuals typically have to give something up in order to gain the benefits of cooperation. This amounts to reaching a consensus or a compromise that agents must make to align their own preferences in order to act in a coordinated manner.[14]

Given that the conceptions of understanding, organization and agency delineated in the Organization Metaphor are conceived with information processing in mind, it is not surprising that we believe they will be well served from a technical standpoint by the constructs of computational models.[15] We are not the first to suggest computational models of intelligent action (e.g., Simon, 1969), it is instructive to note the compatibility of the Organization Metaphor with other contemporary theories. Our systems science point of departure and our commitment to recursively defined structures made it inevitable that we would develop a conception of cognition that mirrors some important aspects of Minsky's "Society of Mind." More deeply, the ineliminable linkage binding individual action, the dynamics of other agents', and the agents' environment, combined with our rejection of a distinction between communication and action,[16] constitute a formal restatement of Heidegger's phenomenological view (1962). This phenomenological perspective has recently been restated in linguistically oriented works such as of Winograd and Flores (1986). Such works purportedly defeat the program of Artificial Intelligence by demonstrating its inability to capture the characteristics of intelligence entailed by Heidegger's phenomenology. In one sense then, one may measure the success of our theoretical and methodological approach to organizational intelligence by evaluating just how well it reifies these phenomenological and hermeneutic aspects of intelligent systems in our technical (and, admittedly, technological) language. Our ability to "make technical sense" of key issues pointed to by phenomenological analysis, will overcome these programmatic challenges.

4. Organizations and their Evolution.

In the preceding section, we used our structural commitment to a unifying model of agents and their environment to deligitimize the distinction between individuals and their organizations. We claimed that intelligent "individuals" are not irreducible unities but rather "organizations of mind." In this way we began to develop a conceptual perspective that rejects the foundational status of a group-individual distinction.[17] This may be a

---

[14]One of the students in our laboratory at Stanford is focusing in these isses (Courand, in preparation).

[15]The technical aspects of our modeling approach will be discussed in Section 5.

[16]Both are forms of information from the perspective of an agent.

[17]Our approach is compatible with those that focus on the "decision" as an entity, e.g., Joel Lawson's claim (at the DTDM 1988 coordination meeting) that group decisions require both high level coordination

provocative departure point for our study. It may even represent a novel perspective for the organization-theory mainstream.[18] However, our perspective does not in fact represent a new technical viewpoint. Systems theorists have been proposing variations on this theme for forty years. However, our contribution will indeed be original if we can reveal the functional "connective tissue" enabling an organization to act as a coherent entity. We can then elucidate our claim that groups have trouble making decisions for the same reason that individuals do. And, we can demonstrate how groups, when they act in a well-coordinated and unified manner, can be "superhuman" (or better, "super-individual"). By revealing the nature and function of this connective tissue, we may eventually be able to clarify the concept of an "group decision," provide insight into what distinguishes a team of experts from an expert team, and eventually develop a normative measure of group effectiveness.

4.1. Agreements: The Functional Connective Tissue.

We have claimed that an individual agents's modifications of the environment based on its understanding provides the basis for communication and inter-subjective agreement among agents. When two agents meet, each brings to the interaction its own particular understanding of themselves, the other agent, and the rest of the world. The effectiveness of these models in guiding an agent's actions reflects the quality of that agent's understanding of the other agent with whom it is presently engaged.

We define *agreements* to be the explicit and implicit protocols that guide these interactions. Agreements can come to define a relationship between agents by defining a relationship between their individual understandings.[19] Agreements determine the commitments made by one or more agents to the interests of others. More specifically, they are *the functional implications of an agent's understanding of its individual commitments.*

Consider the generic interaction of two agents depicted in Figure 1 below. In developing understanding, perceptions associated with the other agent are organized by each agent in relation to its current belief and goal structures. Each agents may take action, based on its understanding, toward furthering its goals. The form and substance of that action will be governed by the agreements that exist -- explicitly and implicitly -- between the agents. Simple conversations in natural language express many common agreements: two English speaking agents will normally agree that objects on which it is appropriate to sit shall be labeled by the term, "chair." Perhaps we can say that, at a higher social level, cultural norms (such as linguistic norms) arise out of complex webs of agreements covering minute details of interaction. Such norms cover a broad range of coordinated action and knowledge, ranging from such concrete acts as how one shakes hands to abstract conceptions such as the nature of authority relationships and how they are maintained. Such agreements, incorporated as "beliefs about the world" by individual agents, contribute to their "social" understanding.

---

of different parts of a decision problem as well as lower level coordination of different agents carrying out a single part -- but rather is simply a more useful framework for holistic, formal analysis.]

[18]Our reading of the literature in this field has shown no examples of theories that reject the distinction between individual and group intelligence in anything like the manner we have outlined.

[19]The reader will recall the technical sense of understanding we developed in the previous section.

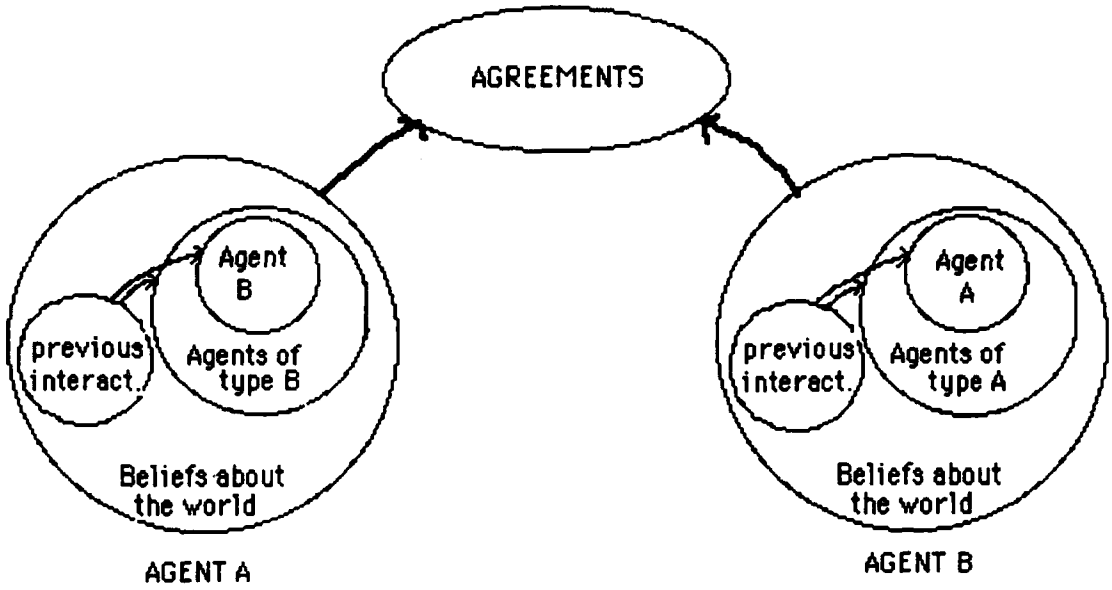


Figure 1. Inter-Agent Agreement.

In many social organizations certain agreements are made more effective by being reified. Often documents such as contracts, by-laws, or constitutions are used to accomplish this purpose. Agreements can also be reified in the decision-making activities of specific group members. For example, in the C^2 domain, a Combined Warfare Commander in a battlegroup operationalizes higher level policy. These reified agreements are seen by group members as something that stands outside of themselves and that define an "objective" social environment in which they participate. To wit, the agreements in figure 1 were deliberately depicted as being outside of both agents.

While such agreement-reifying entities as contracts, constitutions, and organizational decision-makers represent some agreements, they do not fully embody them. An organization may even have reified a set of agreements that belie the real arrangements that constitute the group's true nature (e.g., the constitution of a totalitarian state). The agreements that define an organization's true identity lie in the totality of that organization's dispositions toward action; they are revealed only in interactions or in the changes to an individual's actions upon entering the organization. In this respect, many organizational agreements are will not be reified as laws, rules of conduct, or codified doctrine and may not explicitly appear in the decision-making behavior of some organizational elements. Because these agreements are transparent, recognizing their structural and functional implications may be quite difficult, but these implication will be clear to, as well as accepted by, the organization's members. Agents entering organizations dominated by unarticulated rules may require long periods of "socialization" to come to understand the agreements according to which they are expected to function.

History is rife with stories in which exploitation of tacit agreements offered tremendous payoffs in terms of distributed C^2. General Patton, for example, made sure that his staff met informally every day and required that each member bring a list of specific questions to him every week. Otherwise, they were largely left to formulate strategy on their own. As a result, his staff came to understand his thinking deeply and were confident of his support. In the final raids on Germany, the 3rd Army advanced more quickly than any army in

history while still engaging in numerous battles. Furthermore, archival records show that the 3rd Army required fewer reports and used top-priority communications channels less frequently than did any other headquarters (Van Creveld, 1985).

The history of the Israeli Defense Forces (IDF) provides examples of both the effectiveness of tacit agreements in distributed command as well as the havoc wrecked when, even with technically superior communications equipment, such capabilities were diminished. In the 1967 war, informal coordination and improvisation among small, largely autonomous fighting units lead to the sound defeat of an Egyptian Army who attempted to eliminate the need for coordination at even the lowest levels. Indeed, the unexpected success of certain tactics lead to revisions in formal IDF doctrine. In 1973, "improved" communication equipment that allowed control to be concentrated in Tel Aviv served only to facilitate bureaucratic infighting among generals. Internal conflict and misinformation resulted in worst losses in IDF history (Van Creveld, 1985).

While greater formal understanding of tacit agreement systems will clearly improve the design of C^2 systems, there is also much that could be gained by studying the agreement structures of very formal systems. For example, rather than firing multiple independent missiles in order to increase the statistical probability of success, one could imagine communications agreements embedded in the control systems of the rockets that allow the missiles to "cooperate" in their attack. This is precisely the type of coordination that is envisioned in the "Brilliant Pebbles" ballistic missile defense scheme.

The Organizational Metaphor and the notion of agreements just described constitute our structural and functional phenomenology for modeling organizations. To close the loop, we must convey the relationship between them. Then, in Section 5, we will begin to formalize some of these notions in a computational model.

4.2. Breakdown and Organizational Evolution.

When agents arrive at agreements that "work," each is able to pursue its own goals without conflict. Breakdowns may occur in which existing agreements do not provide sufficient guidance for coordinated, cooperative actions. It is during these breakdowns that the need for new agreements becomes apparent. In this way, the organization evolves, and, over time, the most enduring agreements tend to become transparent. This latter fact contributes significantly to the difficulty in analyzing group interactions and defining organizational identity.

Agents recognize breakdown by noting disparities between the actual behavior of other agents versus what is predicted by their internal model. We have identified two distinct forms of breakdown to date. The most straight-forward form of breakdown occurs when agents' individual goals are mutually compatible but their models of the "world" are different. Differences in their ways of organizing their experience lead these agents to draw different "meanings" from the situation at hand. Such information disparities are the root of situation assessment issues in organizational contexts such as C^2 or institutional management. They have traditionally been addressed through improved communication systems and research into methods of "information fusion."

The second form of breakdown involves value conflicts of two forms: (1) a previously unrecognized conflict, or (2) lack of coordination of objectives among multiple agents. In the first sub-case, agents often find themselves with incompatible objectives that stand in the way of carrying out their proper roles in an organization. For example, members of a

C^2 organization may find themselves competing for resources to the detriment of overall organizational effectiveness. In the second sub-case, agents may recognize a previously unnoticed possibility to collude for mutual benefit. A ship under attack might relay the incoming torpedo bearing to a friendly submarine in order to help it locate and destroy the enemy boat. In either situation, agents may implicitly change an agreement unilaterally or may explicitly confront other agents with the problem and seek to explicitly improve coordination. Agents achieve mutually compatible models of each other by making agreements; this is the deeper sense in which they mutually "reach an understanding".

The computational modeling research discussed in section 5 has been driven very significantly by the desire to understand how breakdown is managed by a problem-solving system. We have aimed to develop a model of problem-solving systems that can properly react to the occurrence of events that signal a needed change in the problem-solving tasks being carried out or in the methods by which those tasks are being completed.

4.3. Summary: Toward Organizational Rationality.

If agreements indeed form the connective tissue that transforms an aggregation of independent individuals into a coherent agent, then agreements must determine the basis for rational action by the organization-as-a-whole. This is perhaps the most important aspect of our research on the concept of agreements. It is certainly the most challenging. Let us summarize our commitments so far.

Our three foundations questions were

1. Why, or under what conditions, do organizations exist?
2. How are organizations as entities related to the individuals that constitute them?
3. What form of intelligence or rationality, if any, can one attribute to organizations?

In answering our first question we postulated that agents do not merely act "rationally." They do so while adapting to constraints on the availability of resources such as time, information, and cognitive capacity that are critical needed in problem-solving activities. Agents exhibit constrained rationality. *Constrained rational agents are driven to organize when, and only when, they believe that the benefits of cooperation outweigh the losses.*

Thus, we have answered the first question by proposing that a rational process of decision-making triggers the process of organization. In this process, agents analyze and resolve the cost-benefit tradeoffs between acting individually and becoming part of a new agent, the organization. Their deliberations about this may take place separately or in some cases together, via consensus formation (Courand, in preparation).

We answered our second question by proposing that organizations are constituted so as to manifest a composite "understanding" (which, the reader will recall, we defined as a basis for *adaptively acting* to achieve some set of objectives). This composite understanding reflects the individual understanding of the organizers, modified so the mutually held objectives can be realized through the coordinated actions of individuals. *Organizations reflect their members' objectives and compromises, and they manifest the power of the members' combined capabilities. There is no mystic "social force" in groups independent of the structure and capabilities of its members.*

Agreements provide the technical mechanism for this coordinated understanding and action. They provide the functional "connective tissue" of the organizations formed on this basis

and embody the constraints on individual action that coordinate the group's members in two fundamental ways. First, agreements determine how and under what conditions information will be shared among the members of the group. This role of agreements serves the basic need of multiple agents to possess a mutually compatible understanding (i.e., basis for adapting and learning) of the world in which their organization functions. Second, agreements align the agents objectives to the extent needed to enable the organization to operate through the actions of those agents as a coherent entity. This role of agreements serves the basic need of multiple agents to coordinate their actions in order to achieve mutually compatible effects, that is, not work at cross purposes.

We can now sketch an answer to our third, and most challenging question. Agreements, whether they are reified or entirely implicit, define the standards for group conduct to which every agent in the group is expected to adhere. Agreements reflect the rational choices of agents about whether, and if so how, to organize to adaptively respond to constraints on their individual action. Agreements, once established, define the organization as a single, coherent decision-making agent. This composite agent, like any other, acts ideally to maximize its expected utility. But, this composite agent, again like any other, must adapt its actions to meet imposed resource-constraints. Therefore, the notion of organizational rationality has a clear meaning in our theory. *By determining the agreements that define the organizations dispositions to act, we can evaluate its effectiveness in meeting the normative standards of constrained rational action* (cf., Goode, 1983; Fehling et al., 1989; Fehling & Breese, 1988).

5. Modeling Organizations Computationally.

This section describes Schemer, a computational model of autonomous problem-solving agents.[20] The problem-solvers modeled in Schemer are capable of reflectively controlling their problem-solving in virtue of the computational mechanisms provided in Schemer plus and a set of decision-theoretic control principles. In exemplifying some of Schemer's features in this section we will occasionally focus on ways in which agent's can manage their attempts to synthesize, and execute, and perhaps modify their plans to achieve various objectives.

Researchers in artificial intelligence have long been interested in the topic of problem-solving control. Some investigators, like Lesser (Durfee and Lesser, 1986), Hayes-Roth (1985), and others (Genesereth and Smith, 1982), have focused on developing general architectures with features that support explicit reasoning about control of problem-solving actions. Our own early research also exemplifies this approach (Fehling, 1982; D'Ambrosio, Fehling, et al., 1987) with development of a general problem-solving architecture that supports reasoning about control especially suited for application to real-time problem-solving tasks. Other research has emphasized that control reasoning exacts a computational price that must be offset by the benefits of such "meta-level" reasoning (Barnett, 1984; Rosenschein and Singh, 1983).

Schemer attempts to more thoroughly reconcile these two research perspectives. We extend earlier efforts to develop a computational architecture that supports control reasoning by incorporating domain-independent, rigorous, and sound principles for control of problem-solving. Though previous approaches to problem-solving control, or so-called meta-level control of reasoning, have provided *mechanisms* by which control information might be expressed (Genesereth, 1982), there has been little emphasis on formal principles

[20]This section presents a revised and extended version of one found in Fehling and Breese (1988).

on which these control decisions might be based. The approach to control of reasoning we have exploited within Schemer (cf., Fehling and Breese, 1988) applies the concepts and principles of mathematical decision-theory. We endorse the view that problem-solving control is a resource-allocation decision problem. An agent managing its actions is constantly making irrevocable commitments of available resources while facing uncertainty about the problem domain. Resource allocation under uncertainty is a topic which has been extensively studied and developed in the fields of decision theory (Savage, 1972) and decision analysis (Howard, 1968). Therefore, we have based our approach to problem-solving control on formal, prescriptive decision-making principles. We do not mean to imply that we advocate that all decision making and planning should be modeled in purely decision-theoretic terms — quite the contrary. We espouse decision theory as a useful perspective stipulating a set of formal standards for the design and performance of autonomous agents and multi-agent groups.

Next, we discuss some important issues and aspects of adaptive problem-solving control in complex, dynamic environments. The remaining portions of this section describe how Schemer addresses these fundamental concerns. We describe the general architecture that allows the interleaving of solution-construction, solution-execution, information-gathering, and belief-refinement activities. We then discuss how normative conceptions of problem-solving control (viz., decision theoretic methods) can be realized within this computational framework. We exemplify the approach in a scenario involving an autonomous robot doing path-planning. These principles we illustrate underlie the *reflective control* of the agent's own inference activities, as well as more standard "goal reduction" planning actions. We conclude with a discussion of important issues on which we are now actively focused and some research concerns that we intend to address in the near future.
The Schemer architecture operationalizes the systems concept we are developing. When implemented and running on standard computing hardware Schemer provides a very powerful framework for conducting experiments and analysis of organizational problem-solving according consistent with our theory of dynamic organizational rationality. It provides a concrete platform upon which various approaches to the theoretical issues just cited may be addressed. It is a framework clarifying the connection between approaches to information disparity problems such as studies of time- and information-limited rationality and approaches to value interactions such as Arrow's Impossibility theorem and other game-theoretic research. Explicating this framework formally and rigorously is precisely what we believe will ultimately serve as a common language for C^2. Because we are shaking C^2 at its foundations, however, the concepts we are formulating must by necessity be quite abstract. In previous reports of this research, we have merely sketched the foundational commitments that we believe will make Schemer and the theory of Dynamic Organizational Rationality an appropriate basis for addressing specific research issues such as limited rationality, resource bounds, and resolution of value conflicts. In this section we now present a brief description of Schemer as a framework for operationalizing and testing these abstract ideas.

5.1. Reflective Control of Resource-Bounded Problem-Solving.

An autonomous agent, operating in a complex and constantly changing environment, combine the ability to react immediately to environmental conditions and, at other times, formulate and carry out plans to achieve the agent's objectives. However, in such an environment these problem-solving activities will be constrained by limitations of critical problem-solving resources such as time, information, and other critical resources. In response to these dynamically changing constraints, an agent must judiciously manage its reasoning and other activities to make the best use of available resources. We refer to

problem-solving under these conditions as *resource-bounded problem-solving* — controlling and adapting problem-solving actions to meet critical, contextually-determined constraints on problem-solving performance.

Resource-bounded problem-solving requires that the agent be *reflective*. That is, the agent must be somewhat aware of its current and past activities and their effects, the agent's future commitments, and the relationship of these factors to potentially critical conditions in the task environment.[21] In the following sections we discuss several important issues which motivate this research. Though not intended to be a complete discussion of critical aspects of limited rational agency, they highlight some particular concerns of our work in this important area.

5.1.1. Information Management — Receiving and processing sensor information.

When operating in a complex, dynamic environment, agents are unlikely to be assured of having complete or timely information about task-relevant conditions. They must inevitably devote some of their efforts to actively sensing the world and interpreting the acquired sensory information. Moreover, the agents' sensory band-width will typically be insufficient to allow them to keep a complete, updated view of their environment, even with this active sensing. Thus, autonomous agents must dynamically decide (or else, embody implicit decision criteria that determine) where and how much relative effort to give to collecting and interpreting new sensory information.

Agents must also passively monitor for the occurrence of certain critical events that signal the need for changes in their problem-solving commitments. Consider, for example, an autonomous space-exploration vehicle, operating in a harsh environment such as the surface of Mars. Such an agent might depend crucially on the ability detect the occurrence of a sudden dust storm so as to immediately protect easily damaged sensors or other devices that might be deployed while performing regularly scheduled tasks. On the other hand, this monitoring requirement would not hold for this same exploratory agent performing the same task on the surface of the moon, since dust storms are not a likely occurrence where there is no atmosphere.[22] As these examples show, the set of critical events for which an agent must monitor are not fixed. They are, at least partly, a function of the agent's current activities and the anticipated features of the agent's task environment.

In summary, the information-management responsibilities of autonomous agents raise important issues for control of reasoning. The relative effort to be given to information-gathering and interpretation is a complex control decision that depends on multiple, changing factors. In addition, agents cannot in general afford to give undivided attention to a narrow, fixed set of concerns. Thus, in carrying out actions agents face another type of control decision having to do with the need to monitor for, and respond promptly to, certain critical events. Unfortunately, little attention has been given by the AI community to these information-management issues.

5.1.2. Action Management — Developing, executing, monitoring and modifying plans

---

[21] The environment to which we refer will most likely include those conditions of the agent's own physical state of that the agent is capable of observing.

[22] However, the agent might now be monitoring for sudden changes in light intensity, the occurrence of a meteor shower, etc.

Based on the understanding gained by an agent's information-management activities agents must determine its objectives and generate plans to achieve them. The detail that agents put into a plan at a given stage of problem-solving depends upon such factors as the completeness of information available to the agent at planning time, the criticality of that information on planning choices, and the urgency of actions that are or will be embodied in the plan. Herein lies an important control decision that has been addressed poorly if at all by AI planning research. Most extant AI research on planning makes the simplifying assumption that planning can be completed before any actions are taken. However, it seems obvious that agents operating in with limited, changing information in realistic environments must typically engage in partial, incremental planning and the interleaving of planning and execution (cf., Fehling et al., 1989).

Having synthesized a plan to some level of detail, agents must then determine when and how to carry out that plan. Central to successful satisfaction of their objectives is the ability to simultaneously monitor the results of executing solutions as well as other relevant conditions in the task environment, and be able to modify existing plans and/or re-plan as conditions in the world change. The activities of planning and execution are coupled in a fundamental way in that the world provides continual feedback to the planner regarding the progress and resolved state of the world. Here again we have exposed issues that extant AI approaches to problem-solving have largely failed to address. In general, because problem-solving models in AI provide no way of interleaving inference activities that construct problem-solution tactics (e.g., planning) with solution-execution and knowledge updating are ignoring a critical aspect of real-world reasoning and action.[23]

5.1.3. Managing Uncertainty

As noted earlier, agents' knowledge of the world is always incomplete and subject to change. That is, agents must cope with varying degrees of uncertainty. To cope they must be able to reason *about* uncertainty as well as carry out reasoning and other actions that are somehow robust in the face of uncertainty. Existing AI problem-solving methods provide only minimal ability for an agent to construct problem-solutions under uncertainty (Tate, 1986; Charniak & McDermott, 1985; Cohen & Feigenbaum, 1982). Uncertainty may be associated with problem-solving information in many ways. Some particular manifestations of uncertainty that arise in realistic domains include:

- Plausible world beliefs — Agents' belief about some aspect of the environment may be only plausible rather than logically definite (i.e., absolutely true or false).
- Environmental dynamism — At any time changes may occur in the environment that (a) are not due to agents' actions, and (b) may affect the validity of their efforts to build, modify, or execute some problem-solution. The agents' knowledge of these independently occurring processes is incomplete.
- Uncertainty of effects — Agents may be uncertain about the effects their own acts that might be specified in a plan.
- Plausible objectives — At a given time agents may be uncertain about the need to take actions to accomplish some particular objectives. It may nevertheless be useful to begin acting in service of some such plausible objectives. For example, agents might realize that, if they wait for certainty, then too much time may have passed for these objectives to be achieved.

---

[23] The work of Georgeff and Lansky (e.g., 1987) might be considered an exception. However, that work does not actually provide a planning method. Instead it offers a strictly sequential control structure for executing and interleaving pre-constructed plan components.

In reflecting on, and managing, these various forms of uncertainty, agents' abilities to recognize and assess their degree of uncertainty fundamentally drives their actions in gathering and interpreting new information or to refine existing beliefs by using various forms of inference. Moreover, uncertainty management activities provide an essential link among information assessment, solution planning, and plan execution. A plan for solving a problem is a potential commitment to action and, therefore, a potential commitment of resources. A rational agent's degree of commitment to a plan should, therefore, reflect that agent's certainty about the information upon which that plan is based. Thus, an agent's reflective awareness of its own certainty about its own problem-solving state as well as the state of the task environment should play in that agent's decisions to actually carry out some problem-solving tactic, as well as in decisions to further refine the tactic or to expend resources gathering additional pertinent information.

Here again, extant research in AI falls short of adequately addressing an important set of issues. AI planning methods reported in the literature, with some exceptions (Doran & Traynor, 1985 ;Drummond, 1985) do not address the need to combine planning, plan execution, with actions to explicitly improve the agent's understanding. Recognition that there are certain actions that can be taken which explicitly reduce the agent's uncertainty, and the consistent evaluation of these options is a deficiency of many standard approaches.

5.2. An Approach to Problem-Solving under Uncertainty.

The preceding issues reveal the significance need for methods that accomplish the reflective control of reasoning in any model of autonomous, rational agency. Intelligent agents must decide how to divide their activities among potential methods for managing beliefs (by collecting information or doing inference), refine their own view of tier plausible present and future objectives and preferences, synthesize or modify their planned commitments to action, and execute these plans at the appropriate time. Agents may also need to choose from among alternative method for accomplishing their objectives in any one of these areas. In carrying out this control function an agent must be able to estimate the relative costs and benefits that accrue to these various choices. This cost/benefit estimation must cope with the pervasive and fundamental uncertainty of the beliefs that hold for any agent operating under realistic conditions.

More succinctly, to cope with the issues we have been discussing, intelligent agents must be able to construct and use general *strategies* for achieving its goals that combine two important types of action. These are

- *Goal-Achievement* actions that directly contribute to satisfying the agent's objectives, and
- *Belief- Management* actions that help to reduce the agent's uncertainty by gathering new information or refining (e.g., through inference) the agent's existing knowledge.

Goal-achievement actions are the types of acts found in plans produced by a conventional AI planner. Typically such a planner uses currently available knowledge and beliefs to construct a sequence of actions each of which directly contributes to the achievement of the stipulated objective. Examples of such actions include; clearing the top of one block before placing it on top of another, moving a chess piece to "pin" an opponent's defending piece prior to attacking the opponent's king, or moving to an intermediate point in a route planning task. The planner's choice of actions is derived from a background state of information, e.g., a knowledge- or belief-base, that encodes the agent's present

understanding of the world. Goal achievement actions are not selected with respect to how they would change that state of information.

In contrast, belief-management actions are specifically intended to change that state of information. These are attempts to actively acquire new, task-relevant information or expend resources inferring additional facts from other explicitly stored facts. It turns out that belief-management activity might require agents to temporarily move *further* from some having achieved primary objectives in some measurable sense. For example, an exploratory robot might determine that its uncertainty about its environment is so limiting as to require suspending its current experimentation (its primary objective) and roam about the area gathering better information about topography, environmental conditions, type and conditions of objects that might be used to aid the robot in its other activities, etc. Although these activities may well detract from the primary objective in a local sense, they have the potential to greatly improve future performance. The decision to undertake belief management actions is non-trivial. If agents are under severe time constraints, for example, then the benefits of information updating may be outweighed by the potential costs of failing to accomplish primary objectives. The presence of uncertainty exacerbates the difficulty of this choice.

Autonomous agents to face a difficult choice between using their current beliefs to complete their primary objectives, or taking explicit actions to improve that background state-of-information upon which effective task performance depends. Agents must carry out a strategy that balances the benefits of carrying out goal-achievement actions that have a chance of achieving objectives in the near term with belief-management actions that have some chance of making general improvements in problem-solving effectiveness in the long term. In the example of planning, agents must decide whether to spend more (or less) time using current knowledge to build and use plans at the expense of opportunities to improve the base of beliefs from which such plans are derived.

The distinction between goal-achievement and belief-management actions is analogous to a distinction in a quantitative, analytic method for adaptive system control known as *dual-control theory* (Feldbaum, 1965; Tse & Bar-Shalom, 1976). A physical system operated according to dual-control uses its feedback information for two alternative (i.e., dual) tasks. First, as in standard control systems the feedback signal can be applied to some quantitative model to improve the stability of the output or make that output converge upon some set point(i.e., goal-achievement actions). Second, the feedback signal can be used to plan ways to gather information that can help the system modify and improve the its stored, internal model (i.e., belief-management). Interestingly, in the latter case the system may need to temporarily de-stabilize its output in order to obtain feedback information that can be used to improve the system's internal model. Analogously, planning agents within the framework described here carry out actions that alternatively move them closer to some goal or improve their understanding of the domain. And, as for dual-control systems, these planners may need to temporarily deviate from approaching a goal in order to acquire model-improvement information. Dual-control techniques are mathematically very powerful and they provide a strong indication that the distinction between goal-achievement and belief-management is well-founded and general. However, as practical methods for controlling real-world system dual-control techniques are computationally intractable; hence, they have not found widespread, practical use. One hope of our current research is to provide a computationally feasible approximation to the powerful, analytic methods of dual control.

Much of our current research focuses directly on these issues of problem-solving control in a resource-limited environment. And, since we strive for an analysis that is both general and based on explicit, rigorous, and justifiable principles, we are combining the results of

our previous research on domain-independent computational architectures for control reasoning with general principles of decision-making under uncertainty. These principles offer a basis for a rigorous, normative theory of reflective problem-solving-control (Fehling & Breese, 1988).

5.3. An Architecture for Control of Problem-Solving under Uncertainty.

The first step in our research has been to specify a general problem-solving architecture capable of supporting flexible, reactive control and coordination of belief-management and goal-achievement actions. By the evolutionary development and testing of this architecture we are evolving a strong computational model of an intelligent problem-solving agent that can adapt its information-gathering and interpretation activities with its actions of planning to achieve primary goals, plan execution, plan monitoring, and replanning. The architecture we have devised, called Schemer, is the newest version of an AI problem-solving framework developed by Fehling and his colleagues over the last five years.

Throughout the course of Schemer's development, this architecture has been tested by being used as the architectural framework for a wide range of problem-solving applications. Schemer has proven especially useful as the problem-solving architecture for systems that must perform acceptably in complex, dynamic environments. Successful Schemer applications have been built for a number of real-time, "process management" applications such as diagnosis or control of complex manufacturing processes (e.g., Raulefs et al., 1987), automated performance management of advanced avionics systems (Keller & Bedoya, 1983; Guffey, 1986), and monitoring and task-management in a distributed information-processing system (Fehling et al., 1984). D'Ambrosio et al. (1987) describe an intelligent, real-time controller for a material-composition process. This system is based on a special version of Schemer called the Heuristic Control Virtual Machine, the HCVM (Fehling et al., 1989a).

Schemer has also been used as the foundation of general knowledge-system development environments[24] for building process management and other "real-time" AI applications. Raulefs et al. (1987) describe the "HPC" development environment that was constructed as a development environment for building a wide class of intelligent process-management applications. HPC uses the HCVM version of Schemer as its basic architecture and inference engine and adds facilities that support development, testing, and modification of applications built using HPC.

These application-development experiences with Schemer have led us to the current Schemer design described below and suggest that Schemer is a superior architectural framework within which to develop an sound approach to control of problem-solving under uncertainty by multiple problem-solving agents. Schemer incorporates several important and unique features that have largely evolved in response to the needs of applications such as distributed, real-time process-management. Such tasks exemplify the various issues of uncertainty management and flexible control of problem-solving discussed earlier. They also require the ability to encapsulate in a single overall framework the loosely coupled, coordinated actions of multiple task specialists. Indeed, our current perspective on, and approach to, the topic of problem-solving under uncertainty has been derived in large measure from extensive experience in using Schemer for such applications.

[24] So-called expert-system "shells."

Dynamic Organizational Rationality

The current changes in Schemer's design are intended to more adequately address a limitation of our earlier work on problem-solving architectures for real-time and other resource-bounded problem-solving applications. In previous research with Schemer the methods for coping with uncertainty and for reasoning about control were ad hoc and application-specific. We have focused on two critical improvements:

1. incorporation of architectural support for encapsulating decision-theoretic methods to achieve the type problem-solving control just described, and
2. modification of architectural features to support the modeling of arbitrary forms of interaction in multi-agent problem-solving applications.

Schemer's design has for some time been quite suitable for modeling multi-agent, distributed decision-making applications such as C^2. However, our more recent commitment to support decision-theoretic methods for resource-limited problem-solving control has forced us to make some major improvements to Schemer. Incorporation of decision-theoretic control methods imposed new requirements on the problem-solving architecture within which these control methods are to be embodied. We wish to emphasize four architectural requirements entailed by this view of control reasoning.

First, an architecture must support encapsulation and interleaved control of multiple, independent, alternative problem-solving methods. An intelligent problem-solver operating in a complex, uncertain context must often make decisions about which of many candidate tasks to undertake at a given time. Sometimes the agent must decide how to interleave the performance of more than one task. Even when dealing with a single task, an agent constrained by time ..., or other resources may need to evaluate the relative costs and benefits of alternative ways of completing a single task in order to determine which is the most promising approach to adopt. Finally, in an uncertain world these control decision may need to be revise in reaction to some critical change in the problem-solving context. To support this view of problem-solving an architecture must support separate encapsulation of, comparison among, and interleaved application of alternative problem-solving methods.

Second, the architecture must provide facilities with which the various, interleaved problem-solving methods and distinct agents can flexibly share information representing uncertain beliefs about the task environment. Problem-solving elements (or agents) that address distinct issues may need to share information. For example, a belief-management component such as a specialized module for interpreting sensor input should produce an updated interpretation in a form that can be used by some other module, such as a path planner. However, in general one problem-solving element may not be able to anticipate which other element is likely to consume this information. An agent may need to choose among alternative ways to solve a single problem and later modify that choice. In this case, too, the problem-solver should be able, at least in principle, to share information among these alternative solution approaches.

Third, the architectural framework must also strongly support the encapsulation and appropriate application of elements that embody the control principles such as those we have briefly discussed. The decision-theoretic approach to control reasoning that we are formulating is based upon domain-independent principles of resource-allocation under uncertainty. The architecture should provide basic structures for representing these principles in a general form as well as instances of these control reasoning principles that are specialized for particular types of application. Furthermore, the architecture should foster efficient use of these techniques for control reasoning.

Fourth, the architecture must provide a fundamental mechanism by which the problem-solving activities, and the control reasoning that shapes them, is responsive to the changing conditions of the problem-solving context. In a complex, dynamic environment a problem-solving agent cannot be entirely certain about past decisions, including decision involving the control of problem-solving. Thus, the architecture must fundamentally provide the basis by which an agent's control reasoning is itself partially controlled by the flow of critical events.

We now present a brief description of Schemer, emphasizing its main functional and structural features that are crucial in providing a proper architectural framework for dynamic control of problem-solving under uncertainty and for modeling cooperative problem-solving.

5.3.1. Schemer Architectural Features.

Experience in building intelligent process-management applications with Schemer has prompted the inclusion of some important and unique features in this general problem-solving model. These features of Schemer have proven essential for intelligently performing tasks in dynamic, complex environments. Such task environments exacerbate the problems of uncertainty we enumerated above. Schemer's problem-solving activities can be flexibly controlled entirely or in part by the flow of events and information in the problem-solving context. Key Schemer features include

• Interruptability	Critical events in the problem-solving context can immediately gain control of Schemer's problem-solving actions, redirecting the problem-solver's attention to respond to these critical events.
• Event-driven, adaptive control	Schemer supports "meta-level" reasoning about control so that problem-solving activities can be flexibly driven by the changing state of events and information in the task environment.
• Modularity	The individual problem-solving elements, both data and procedures, within Schemer can be fully encapsulated in a uniform manner, providing a basis for individuated structure, mutual consistency, and independent transactions for these elements.
• Information sharing	Schemer provides the means by which a group of possibly heterogeneous, individual problem-solving components can share information.
• Multi-tasking support	As a result of its facilities for flexible, adaptive control and modularity we

have found that Schemer provides the ability to schedule and control multiple, simultaneous, and perhaps independent lines of problem-solving activities that must often be carried out by agents acting in complex, dynamic contexts.

These features of Schemer are essential for the approach to control we are developing. We now provide a brief technical description of this architecture's key features. First, we consider the architecture's overall functional organization. Following that we describe the flow of information and control in Schemer.

5.3.2. Functional description of Schemer.

Schemer combines features of blackboard architectures such as Hearsay-III (Balzer et al., 1980) with strong modularity and support for hierarchical classification and feature inheritance normally found in object-oriented systems. Schemer is primarily concerned with dynamically managing the flow of control and data in sophisticated, "real-time" reasoning systems. To do so, it recognizes two fundamental kinds of control regimes: event-driven invocation of reasoning procedures and direct procedure invocation. Unlike traditional blackboard systems, Schemer makes no a priori commitment to one of these styles of control over another. Instead, both of the invocation mechanisms are potentially usable in a problem-solving application. One advantage of this approach is flexibility of expression -- when the exact flow of control for some part of the system is well understood and predictable, it can be explicitly encoded using the direct invocation (often called synchronous) approach; when the exact flow of control cannot be pre-determined, or is otherwise unknown, procedure invocations can be specified as responses to the occurrences of patterns in stored information, yielding the event-driven, or asynchronous, style of activation commonly found in blackboard systems.

Schemer uses agenda-based scheduling methods to manage the flow of computational activities (either synchronous or asynchronous). Modules invoked either synchronously or asynchronously may be placed on the system's *Schedule*, assigned priorities, and prioritized for immediate or eventual execution. This approach adds a layer of overhead processing to the system; however, it also provides the ability for a system to intelligently manage its own resources by monitoring its activities, determining which activities have the highest priority, and dynamically resetting priorities when appropriate. In domains where there are often more computational tasks to be performed than there are available resources, and for tasks larger than some minimum grain-size this overhead layer provides the capability for a system to continue performing reliably and responding to external events even when it is computationally overloaded.

Figure 2 below illustrates Schemer's architecture. This figure depicts the major functional components of the architecture. Before beginning our detailed discussion of this system, we wish to point to two overriding and long-standing objectives that have guided the evolving design of this AI architecture. Careful adherence to these objectives have helped us select the best opportunities for using Schemer to build prototypes and fully fielded real-world applications. These objectives have also guided our ability to exploit this application-building experience in refining and evolving our architectural design work. As result of careful attention to these objectives over the history of Schemer research and development activities, this framework now provides the most mature computational model of which we are aware that simultaneously addresses the following concerns:

- This architecture is the only AI architecture of which we are aware that has been specifically designed *a framework for building resource-bounded problem-solving systems*, with particular attention to *real-time* applications.
- Schemer is also the only AI architecture of which we are aware that has been specifically designed as *a framework for building "multi-agent" problem-solving systems*. The architecture provides a generic, uniform basis for modeling complex interactions among multiple problems solving components, of arbitrary complexity, including components that themselves are instances of independent, communicating, intelligent problem-solving agents.

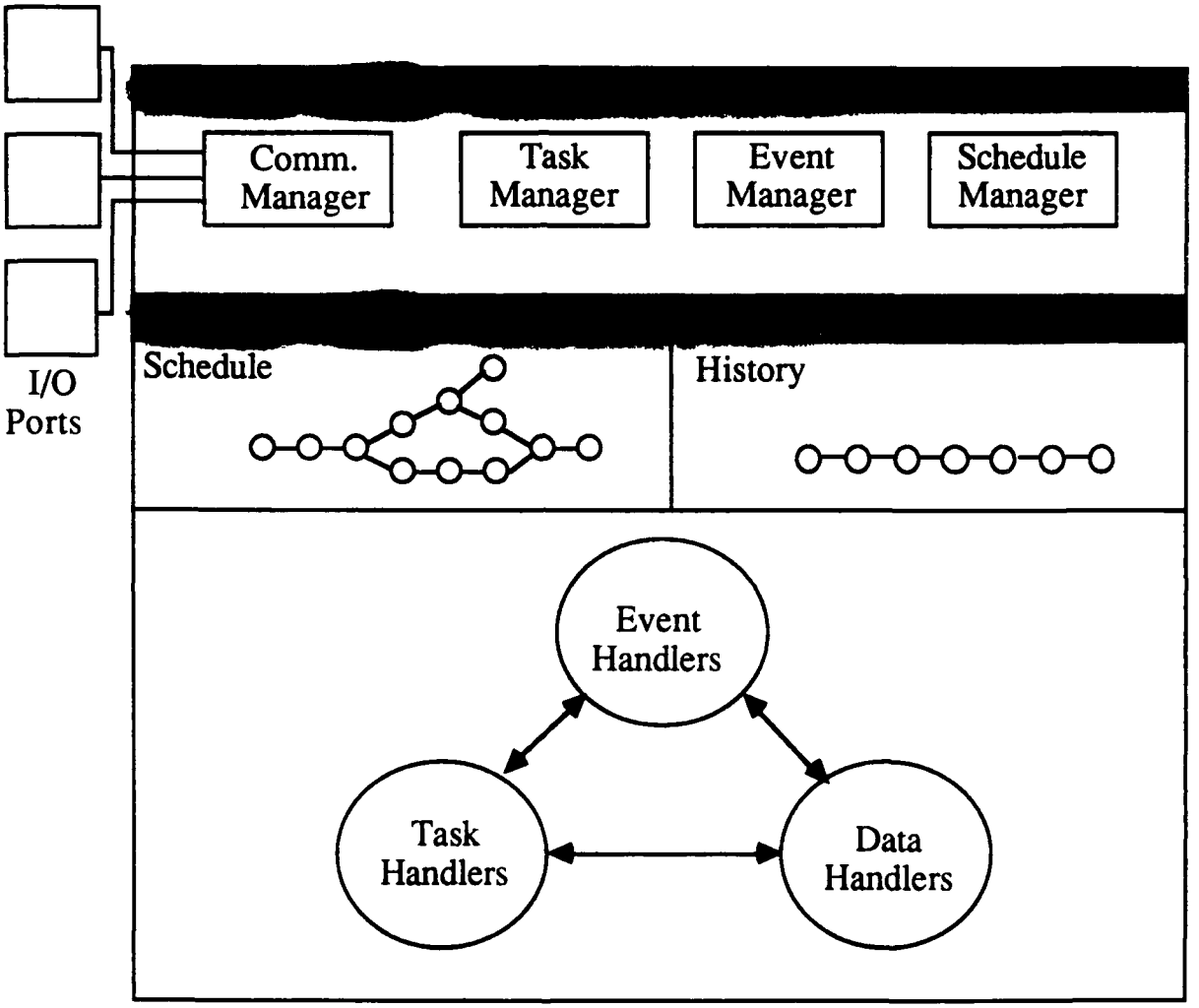


Figure 2. Functional Diagram of the Schemer Problem-Solving Architecture.

We now describe this architecture in more detail. We begin by describing its structural organization. Following that discussion we describe how computation and communication is carried out in Schemer.

There are three major components of this architecture:

- A collection of modular problem-solving elements called *handlers* that represent both the knowledge and other system problem-solving data and the procedural elements that create and refine this knowledge,
- A global store called the *Knowledge Space* that contains the handlers for a Schemer implementation as well as a current *Schedule* of pending tasks (a prioritized representation of potentially executable handlers) and an audit trail, called the *History*, of previously executed tasks,
- A Top Level Controller that controls system-wide activity and is itself made up of four elements:
  - » A *Communications Manager* that processes the flow of data between the knowledge space and Schemer's connections to outside systems via its ports,
  - » A *Task Manager* that runs the next scheduled activities and manages its completion and possible suspensions and later resumption if an interrupt occurs,
  - » An *Event Manager* that operates a distributed triggering mechanism that determines the procedures to schedule for execution in response to the occurrence of certain data patterns in the knowledge space, and
  - » An agenda-based *Schedule Manager* for determining which of the triggered procedures are actually executed and when.

We begin by discussing the handlers. Both procedures and data are encapsulated in Schemer in *handlers*. There are two types of handlers primarily intended to encapsulate procedures: task-handlers and event-handlers. Task-handlers and event-handlers are identical in all respects except the way in which they are scheduled for execution (see below). Roughly, event-handlers respond to pre-specified patterns of data (whenever, that pattern appears, the event-handler is scheduled for execution), and task-handlers are explicitly called by other handlers. We refer to *handlers* when they are being defined and when they are invoked; we often refer to *tasks* when a handler is being scheduled or executed.

The body of a task- or event-handler (an executable procedure) may be encoded in any of several ways; currently, the options are (1) as a set of inference rules, (2) as a Common-Lisp expression, (3) as a "program" of invocations of other task-handlers, or (4) as another, embedded Schemer. The intent is to allow maximal flexibility of expression within the handler while at the same time maintaining a well-defined interface specification among the various modules of the system (i.e., between various modules and the rest of Schemer) which includes a description of how the module is to be invoked.

Event-handlers respond to patterns of data in the knowledge space. The triggering mechanism maintains a record of the pattern associated with each event-handler (stored in a structure that we call the *trigger table*) and the current state of the pattern (whether the pattern matches the current state of the knowledge space). The trigger table's records are updated to reflect changes to the state of the knowledge space. Periodically, the trigger table is scanned to determine which patterns have become asserted (i.e., match some component or set of components in the current state of the knowledge space), and the event-handlers associated with matched patterns are scheduled for execution. When an event-handler's pattern becomes true, that event-handler is said to have been *triggered*. The current trigger-table implementation has been optimized to eliminate two common sources of redundant computation: (1) redundancies among patterns that result in multiple computations of the same pattern or pattern part and (2) recomputation of a pattern whose constituents (the data on which it depends) have not changed.

All computational activity involving task- and event-handlers is mediated by an agenda-based scheduler. In practice, this means that requests for execution of a handler are posted to the *Schedule*, assigned a priority, and selected for execution from the *Schedule* in priority order. Agenda-based scheduling gives this system the ability to manage its own resources dynamically, and it gives it a natural *interrupt* mechanism. Any task that has been scheduled for execution (i.e., is on Schemer's Schedule), can be pre-empted, re-ordered, or even removed at any time that a more urgent task requires the system's processing resources.

The data representing a Schemer system's problem-solving knowledge and beliefs are encapsulated in *data handlers*. These data objects are potentially accessible by any of the other handlers in the same Schemer system. In a manner directly analogous to an object-oriented system, a data handler encapsulates a particular data structure and the operations that manipulate it. Other system components operate on a datum by invoking the handler's internal operations.

Unlike most blackboard architectures Schemer's handlers, including the event handlers,[25] may have local persistent state. The data that is local to a Schemer task- or event-handler is encapsulated in its internal *Data Space*.

5.3.3. Flow of Information and Control in Schemer.

Having described the major functional units of Schemer, the flow of information and control among these components is straightforward. Conceptually, the overall execution cycle is handled by four components: a Communications manager (CM), a Task Manager (TM), an Event Manager (EM), and a Schedule Manager (SM). These elements comprise the basic operations of the top level controller (TLC). For efficiency considerations, the current versions of Schemer implement these four functions in an iterative, sequential manner. In this sequential version of Schemer an activity cycle corresponds to executing the TLC once, which in turn executes each component sequentially in a fixed order. Figure 1 displays these components of the TLC. For purposes of this discussion we may assume that the execution order on a single cycle is CM-TM-EM-SM.

All external communication is mediated by the CM. It has special data structures, called communication ports, or just ports for short. Ports hold data being communicated to or from the system. Thus, the ports define the interface between Schemer and the outside world. A port may be either input-only, output-only, or in-out, thus defining the corresponding restriction on direction of data-flow for that port. Conceptually each port is linked to one or more external objects capable of sending or receiving information. These links represent the channels of information flow between the Schemer system and the object(s) to which the ports are linked. The CM can dynamically create new ports and create or delete links with the cooperation of the external objects connected to these links.

The CM also actively controls the communication actions on behalf of its system's components. On each cycle the CM will collect new data from input or in-out ports and put data to be passed outside the system into its output or in-out ports. Then the CM will carry out explicit communication actions between these ports and the external objects to which they are linked. The way that the CM manages communication for each port depends upon whether that port supports a synchronous or asynchronous style of communication. Synchronous (asynchronous) ports may only be linked to external objects supporting the same style of communication. These two styles of communication are managed differently

[25] Event handlers are analogous to the so-called knowledge-sources or knowledge-specialists in a blackboard system such as Hearsay-II (Erman et al., 1980****), or BB1 (Hayes-Roth, 1985****).

by the CM. A CM cannot complete a send operation being at one of its synchronous output (or in-out) ports until the objects linked to that port complete their receive of the sent data. Similarly, a CM will initiate a receive operation on a given cycle for any synchronous input (in-out) port for which a linked sending object has initiated a send to that port. In contrast to synchronous communication, the CM manages its asynchronous ports without regard to the state of communication of the objects to which these ports are linked. Currently the asynchronous ports are all in-out ports implemented as queues, which means that they can provide or accept data at any time even if these data are not immediately consumed or have been produced at some previous time, respectively. These ports and their management by the CM provide the Schemer system the ability to communicate asynchronously with other independent devices such as users, other computers, or the ports of other Schemer systems in a multi-processing implementation. Newly received data on a cycle will be passed by the CM into the knowledge space of the Schemer system, usually to be further processed by handlers within that system.

The TM selects tasks from the Schedule in descending priority order and executes them. In single-processor implementations of Schemer, the TM selects one task for execution per TLC cycle. In a multi-processor environment, the TM distributes available processors to scheduled tasks again in descending priority order. The TM's main activity is to monitor scheduled task performance according to three components in the task being run: its precondition, its code-body, and its failure-body. Each task, encoded as either an event- or task-handler, will contain an expression called a precondition. The precondition is an expression representing some assertion that is must be "true" (i.e., contained in the knowledge space) for the task to actually be executed. At the moment that a task is selected by the TM to begin execution the precondition of its handler is evaluated. If it evaluates to **true**, then the code-body of the handler is executed; if it evaluates to **false**, then the failure-body is executed. The use of preconditions allows us to distinguish three levels of control reasoning about a task -- all the task- and event-handlers in the knowledge space represent processes that are potentially executable; scheduled tasks (i.e., already triggered or synchronously called) represent candidates for immediate execution; the task(s) selected by the TM represent the system's current execution commitment.

Schemer's approach to task management supports "objective-directed" as well as event-directed control of actions. As in blackboard systems, the trigger expressions of Schemer's event-handlers express the way that changes in the system's data-state partially controls the execution order of the system's tasks. Schemer supports this standard form of event-directed control. Schemer is also able to control the effects of its actions in terms of how well these actions achieve stipulated objectives. This system is also able to find and then schedule tasks that achieve posted goals, even though those tasks have not been triggered in the event-directed manner. By objective-based control we mean these latter two mechanisms. Let us clarify how this type of control is accomplished.

Task- and event-handlers may contain an expression called a *postcondition*. The postcondition represents a partial description of the knowledge space state that is expected to hold after the task is executed. A postcondition can be thought of as representing the purpose or objective that is supposed to be satisfied by execution of the handler's code-body. Schemer uses these postconditions to achieve both aspects of objective-directed control as defined above. First, the TM can use a task's postcondition to determine whether that task has accomplished the effects represented by its postcondition or has failed, allowing the TM to react appropriately in either case.[26] The TM can encapsulate the

[26] Note that this effect cannot in general be guaranteed to hold, since the consequences of a task's execution is determined in part by conditions caused by other processes that hold before and during the time of its execution.

effects of executing a task in a newly created data-handler as a kind of transaction encoding. Any other changes occurring while the task is executing are also encapsulated in the special encapsulation. After the task is completed the postcondition is checked. If it is satisfied then the effects of the task are propagated to the knowledge space and the special encapsulation is discarded. Otherwise, the TM posts a message to the knowledge space that the task has failed, includes the encapsulation, and continues. In this way unanticipated effects of actions can be controlled and sometimes completely avoided.

Postconditions also provide Schemer with a means to carry out goal-directed problem-solving. Special event-handlers, called goal-handlers, respond to the posting of a goal by searching for some handler whose postcondition includes the goal specification. If such a handler is found then the goal-handler passes the discovered handler to the TLC for scheduling and eventual execution. In turn this newly-scheduled handler may post its own goals in addition to (or instead of) carrying some sequence of executable steps. These (sub-)goals then evoke a goal-handler to repeat the goal-directed evocation process. This mechanism continues until tasks post no new goals or some goal is posted which no handler expresses in its postcondition.[27]

By supporting goal-directed control we have added a fourth level of control commitment to the three previously mentioned. A goal-handler whose execution is evoked by the posting of some goal represents a form of partially specified strategic commitment. I.e., the system is committed to dynamically determining, scheduling, and then eventually executing some sequence of actions to achieve a stipulated objective. The four levels of control commitment we have now described are analogous to a similar stratification described by Hayes-Roth (1985) in terms of the BB1 architecture.

The EM acts on behalf of the event-handlers by monitoring for the occurrence of conditions (viz., partial descriptions of the state) of the knowledge space that are match the trigger expression of any of the event-handlers. It is important to note that changes to data-handlers stored in the knowledge space are made only by the CM processing of new inputs from the ports or by the TM's execution of tasks. The EM monitors the changes caused by CM- or TM-controlled activity and updates a special data-structure called a trigger-table. This trigger table is a hierarchically structured representation of the trigger expressions for each existing event-handler and the sub-expressions for each of the full trigger-expressions. The table is annotated with indications of the (sub-)expressions that have been satisfied at a given time. When a full trigger-expression is marked as satisfied the EM is able to determine that invocation records of handler(s) associated with that expression should be passed to the SM for inclusion in the Schedule. These invocation records are called carriers. In sum, the EM is the set of procedures that updates the trigger table, scans the trigger table to determine which patterns (called triggers) are matched in the global knowledge space, and determines which event-handlers should be scheduled for execution. The actual event- or task-handler is not passed around for execution. For each handler to be scheduled the EM creates a *carrier* containing the identity of the handler whose code-body or failure-body is a potential task and other information specific to this invocation such as parameter bindings. The EM supports edge triggering; i.e., an event-handler is scheduled for execution when its trigger condition becomes matched.

The SM maintains the order of the potentially executable tasks on the Schedule. On each TLC-cycle the SM adjusts the priorities of the current members of the Schedule. This activity is called aging. Many alternative aging schemes could be used. In most Schemer

[27] These mechanisms for objective-directed control of action were originally implemented in an early version of Schemer (e.g., Keller & Bedoya, 1983). This mechanism appears to be a generalization of Georgeff's so-called reactive planning system, PRS (Georgeff, 1987).

implementations the SM carries out aging according to the policy that the longer an event-handler has been scheduled for execution, the higher its priority. After aging the SM determines a priority for the newly triggered event-handlers, and adds them to the Schedule. As we noted above, the handler itself is not added to the Schedule; rather, an activation record called a carrier is posted to the Schedule. This approach protects the handler from changes to its local data between the time that it is triggered and the time that it executes, and it allows for the case in which one event-handler is scheduled for execution multiple times with different activation values.

The SM supports a rather flexible representation of prioritization information associated with each scheduled task. Priorities have two descriptors: *level* and *amount*.. A priority consists of one *level* and one *amount*, e.g., <High, 3>. The *level* descriptor has four possible discrete values: *extreme*, *high*, *ordinary*, and *low*. The *amount* descriptor is of type real. *Levels* have precedence over *amounts*. Currently, the aging procedure increments a priority's *amount* by 1, leaving the *level* unchanged; thus, aging increases priority within a *level*, but not among *levels*. The range of values of either *level* or *amount* can be as simple or complex as is required by the specific application.

The functioning of the SM is one important way in which a Schemer system can "reason about its own control." The SM can enact any general approach to scheduling, such as the decision-theoretic approach we describe below as long as that approach is suitable under all the problem-solving conditions that the system will encounter. We have found Schemer's ability to support pre-emptive scheduling to be of particular significance when using Schemer for various real-time applications. The SM can dynamically revise the Schedule on any cycle. This allows the SM to pre-empt some sequence of problem-solving steps with another in response to the occurrence of some "critical event" that has caused the EM to trigger a task of higher priority than any currently executing task. This occurs if a change to some data-handler in the knowledge space (perhaps caused by input via the CM) causes a set of actions to be triggered that the SM recognizes as higher priority than the current actions.[28]

In addition, the SM's scheduling capabilities may be complemented by the use of "control-knowledge" handlers. Such handlers respond to patterns in the global knowledge space, especially the Schedule and History-List structures which are themselves part of the knowledge space. When executed these control handlers can carry out control operations by directly manipulating the Schedule. One important advantage of this approach is that the top-level controller can remain quite simple, and hence efficient, by relying upon control handlers to distribute control knowledge. By distributing control knowledge a Schemer application can flexibly apply alternative methods for (re)scheduling in response to the occurrence of particular events during problem-solving. On the other hand, when an application requires uniform application of a fixed method for scheduling, that knowledge can be efficiently encoded in the SM. And, of course, these two approaches to control reasoning can be mixed.

5.3.4. Schemer as a Framework for Decision-Theoretic Control.

---

[28] In previous versions of Schemer the four TLC components operated in sequence exactly as we have described. Thus, the SM's ability to interrupt current activity was limited by the size of the task currently being executed by the TM. In the current system we are now implementing the four TLC activities take place concurrently. Using this approach we are now able to interrupt one of the TM's tasks while it is being run if the SM notes the scheduling of some higher priority task during that time. In this case the SM generates a true interrupt of the TM, prompting the TM to save the control- data-state of its currently running task, suspend that task, and resume the interrupting task.

Schemer provides some unique and important features to support flexible, reactive control of problem-solving. In particular, the architecture supports a wide variety of techniques for flexible, dynamic scheduling, the ability to employ special-purpose problem-solving modules that can be invoked like any other module and can modify the system's control state, and, perhaps most importantly, true pre-emptive control providing the problem-solving system the ability to react promptly and re-focus its attention in response to the occurrence of critical events. Nevertheless, our work on this architectural framework provides only part of the solution to the problem we have defined -- stipulation of a general computational model of the intelligent control of resource-bounded problem-solving.

Although Schemer's design provides a very robust computational framework for applying appropriately chosen techniques for control-reasoning, this architecture does not by itself offer any such techniques. In order to make this architecture a suitable model of a resource-bounded problem-solving system we must incorporate within its computational framework methods for control reasoning that are based upon sound, domain-independent principles. Our current research is focused in a significant way upon these issues. Let us evaluate Schemer's features in light of our earlier remarks regarding the architectural requirements entailed by adopting such an approach. The reader will recall that we noted four basic requirements: a problem-solving architecture must support

- Encapsulation and interleaved control of diverse problem-solving methods,
- Information-sharing among independent, possibly heterogeneous components,
- Architectural support for both domain-independent and specialized control reasoning, and
- Critical-event driven control of control reasoning itself.

5.3.4.1. Encapsulation and interleaving of diverse problem-solving elements.

As we indicated in our earlier discussion, the Schemer architecture must support encapsulation and interleaved control of multiple, independent, alternative problem-solving methods. Schemer's handlers, with their object-oriented modularity, meet this requirement by providing a discipline for encapsulating each problem-solving element as a distinct type of object. Each handler can encapsulate a specialized type of problem-solving skill. Handlers provide convenient data structures that encapsulate and provide to the TLC information that may be needed by the TLC's Schedule Manager for making control decisions. Distinct handlers can be comparatively evaluated and independently managed to make control decisions.

A handler provides basic features that support a strong distinction between the information that is strictly local to a problem-solving element and information that is to be shared with other elements. A handler provides the the capability for hiding local data and providing a transaction context for protecting the consistency of this local information when other handlers may attempt to modify that data in carrying out their own actions. This feature is essential when multiple problem-solving activities must be interleaved or when some task must be temporarily suspended in order for a more important task to be accomplished that might attempt to manipulate information being modified by the suspended task. Finally, the ports for a handler provide the means for defining a clean interface between each problem-solving element, the other elements with which it directly communicates, and the general control structure (i.e., the TLC) of the containing architecture.

5.3.4.2. Information sharing among heterogeneous problem-solving elements.

We also noted that Schemer must provide facilities with which the various, interleaved problem-solving methods can flexibly share information. This requirement is somewhat at odds with the preceding one which mandated strong modularity, information hiding, and structured communication. In order for the system's overall objectives to be met, the elements that enact diverse problem-solving tasks may need to share information in ways that are difficult to determine in advance.

For example, a system doing intelligent control of an autonomous vehicle might employ a slightly-modified version of a conventional deductive planner to construct a sequence of goal-achievement steps until critical uncertainty is detected. At this point control might be given to another problem-solving element whose job it is to reduce the critical uncertainty by performing some belief management activity (e.g., further assess available information or gather and assess new information). Information derived while planning or the partial plan itself might be useful to the newly invoked module. Once the world modelling information on which planning depends is so augmented the plan must be available for possible further refinement by another module, and so on. Alternatively, the belief-management element may merely augment the partial plan by proposing information-gathering steps to be interleaved with a contingent at the appropriate point in the path-plan (Warren, 1976).

It is important to emphasize that, in cases like the preceding example, one may not assume a determinate relationship between one problem-solving element and another. In this case the agent may have various methods for belief management, each differing in its costs, the quality of its result (Lesser et al., 1988), and other factors. Thus, the intermediate results of the planner must be available to for use by other problem-solving elements that cannot be determined a priori. Similarly, the results produced by the belief-management module selected might eventually be made available to the planning module that was suspended or perhaps to another planning module that could be selected in place of the original one.

Schemer supports cooperative, loosely-couple interaction among multiple problem-solving elements by embedding the collection of problem-solving elements in a global, shared knowledge-space. The various problem-solving elements (e.g., planners, situation-assessment modules, etc.) share their intermediate or final results in the knowledge space in a common format. Thus, the knowledge space represents the common background information that all problem-solving activities can exploit. We are implementing a knowledge-representation calculus that can be used as a basic expression language in terms of which Schemer's modules can communicate. We have also begun incorporating conventional AI planning methods (based upon such systems as NONLIN - Tate, 1976; Tate & Whiter, 1984 and DEVISOR - Vere, 1983) so that the actions of these planners may be properly integrated with belief-management methods we are developing.

5.3.4.3. Support for both generic and specialized control reasoning.

It should be clear from our brief description of Schemer that this architecture directly supports the encapsulation and use of problem-solving components that deal with the problems of "meta-level reasoning," an intelligent agent's problems of control reasoning. This architecture provides two important ways in which control principles can be encapsulated. The Schedule Manager encapsulates a system's basic, situation-independent method for choosing the execution order of task specified by the Event Manager. Alternatively, control decisions can be made by executing specialized "control handlers." These event- or task-handlers respond to the occurrence of specific scheduling problems as represented by the contents of the Schedule. When executed a control-handler applies its particular method for resolving the scheduling problem.

The decision-theoretic methods we describe in section five could be used by the SM in place of the ad hoc scheduling approach we have employed in previous incarnations of Schemer. In previous implementations the SM has used a simple pre-emptive, priority-based scheduling discipline. In this approach the carriers representing potential tasks have an initial priority prescribed by the system developer as a feature of their associated handler. On each cycle of the Top-Level-Controller the current priority of each task remaining on the Schedule is then "aged" (viz., has its priority value modified) in some equally simple and application-dependent manner. The SM's method for scheduling would be made more general and well-founded if we replace this ad hoc approach by prioritization and aging schemes based on analytically sound principles for allocating computational resources, i.e., for dynamically assigning and updating task-priorities based on an analysis of the estimated costs and expected benefits of each task.

A potential problem with this approach is entailed by the computational (and other) resource requirements associated with this method of reasoning about control (Rosenschein & Singh, 1983; Barnett, 1985). The activities of the SM are an "inner loop" in Schemer's overall computational activities. If the computation costs required to explicitly perform a full-blown cost/benefit analysis on each cycle are too high, then they will outweigh the value of this control reasoning no matter how formally sound and general it is. Thus, it may be necessary to restrict the real time estimations performed by the SM on each cycle in response to limitations such as time deadlines. In extreme cases, it may even be necessary to abandon such a method entirely in favor of some simple prioritization scheme like the one we have used in earlier versions of Schemer, however the priorities could be derived from off-line decision-theoretic analyses and considerations.

Schemer's control-handlers provide a possible means of coping with this problem of the cost of control. Rather than performing complex scheduling decisions on each cycle as a part of the SM's actions, control-handlers can encapsulate control-reasoning methods that can be applied (e.g., triggered and immediately executed) on demand, or periodically, or both. This approach has the additional benefit that multiple, alternative methods for control reasoning can be simultaneously encapsulated, each methods being called upon in response to distinct scheduling problems and contexts.

5.3.4.4. Critical-event-driven control of reasoning.

One of the most important objectives in the evolution of the Schemer design has been to fundamentally support problem-solving processes whose control is responsive to critical changes in the problem-solving context. A problem-solver dynamically formulating and executing solutions to problems in an uncertain environment must be able to react promptly to the asynchronous occurrence of such critical changes. In response to such changes, the problem-solving system may decide that its current actions are no longer the most preferable activities to engage in. We have found the capacity for "interrupt-driven" control of problem-solving to be paramount when using earlier versions of Schemer in problem-solving applications that must exhibit reactive, real-time performance.

In this architecture the occurrence of some critical event can initiate a response to immediately interrupt execution of the currently scheduled problem-solving tasks, suspend them gracefully, and commence tasks that are more appropriate in response to the changed information about the problem-solving context. This is readily accomplished by the use of special event-handlers that carry out these actions in response to pre-defined critical events. This aspect of Schemer's design is a natural evolution of the mechanisms for "opportunistic control" typical of blackboard systems such as Hearsay-II (Erman et al., 1980).

In summary, Schemer provides a sophisticated architecture for modelling resource-bounded problem-solving. Schemer provides a number of unique computational features not found in other AI problem-solving architectures. We have implemented an initial version of Schemer in Common Lisp. This version runs on Symbolics 3600 series computers, Xerox 1100 series computers, and on a Macintosh-II using Allegro Common Lisp.

5.4. Organizational Modeling.

The preceding design description has already pointed to those features of Schemer that make it appropriate for modeling and analyzing distributed, multi-agent problem-solving systems. It should be clear from the discussion that this framework provides strong support for modeling problem-solving systems composed of multiple elements whose activities are independent or loosely coordinated. No restrictions are placed on the simplicity or complexity of the contents of a handler in a Schemer system. Hence, these constituent problem-solving element can range from functionally trivial processes to components that model intelligent agents capable of reasoning and acting independently, cooperatively, or competitively in a multi-agent setting. However, we underemphasized two aspects of Schemer's design that makes this type of distributed modeling so convenient, Schemer's formal communication model, and the "self-embedded" nature of this architecture. We briefly address these issues now.

First, Schemer is unique in providing an explicit communication model that precisely describes the way information is exchanged between a Schemer system and its surrounding environment. This communication model is an extension of Reid's (1980) formal model of communication and control relations in computational systems. It is abstract and completely general and thus provides a language with which to model how a Schemer system and its embedding environment effect each other through their actions. This model rigorously provides a precise, formal interpretation of the communication relations that are modeled within it. Both synchronous and asynchronous communication relations can be modeled. Within each of these classes of communication protocols the Reid model supports a full range of distinctions regarding the way in which messages are retained or destroyed in a communication action.

In this model, abstract entities called *ports* (as described in the review of Schemer) are associated with a computational process. A port can be either a read-only port, a write-only port, or a read-write port. Ports are connected to the ports of other entities by formal *links*. A port-link-port relationship established a communication channel between the processes that each own a port in that relationship. As we have already noted, the communication over this link can be either synchronous, meaning that the write (or message sending) event can complete if and only if the read (or message receiving) event has completed, or else asynchronous, meaning that the read and write events are not coordinated. Reid (1980) proves that these primitives are sufficient for formally describing a general class of communication relations among computational processes. This includes broadcast as well as point to point communication.

Perhaps one of the most important features of this model is its ability to represent dynamically changing communication relationships. The Petri net (cf., Peterson, 1981) is a more widely used alternative to the Reid model for describing communication relations in complex systems. In thinking about organizational modeling we were struck by failure of petri net models to readily support modeling of dynamically changing communication channels in an organization. One will seldom find static and unchanging communication relationships in a typical organization. This is particularly true for the types of systems one

finds in C^2 problems. Hence, in contrast to many who are investigating the formal properties of communication we choose the Reid's approach over the petri net formalism.

Schemer's recursive definition of a problem-solving system makes this model very useful in representing the most complex organizational structures. The definition of a Schemer system is very abstract. It allows us to recursively define the components of a Schemer system to be embedded, specialized versions of the same general system definition. For example, the description of "handlers" we presented above focused on the special properties of these elements in there role as constituents in an encapsulating Schemer framework. In fact, these handlers are special instances of a general Schemer architecture. That is, a handler is just a Schemer architecture that is embedded as a component in a containing instance of that same architecture. Of course, each of these instances is specialized with the appropriate features to play its correct role. Also, some set of primitive Schemer instances must be defined to provide the base cases for this self-embedded system specification. This degree of abstraction and self reference in the formal specification of a Schemer application has made this computational model very useful in specifying very complex computational systems, including massively parallel systems and even hybrid systems (cf., Fehling & Wilber, in preparation, for a detailed discussion).

6. Control: Measuring group effectiveness.

During one of the annual DTDM review meeting, one participant from M.I.T. stated that the organizational design problem was not "analytic." Given that this was generally accepted by all participants, it is no wonder that members of the DTDM steering committee decried the continual use of "optimality" as a principle of analysis. Of course, although C^2, and most other organizational analysis problems, may not be analytic, it may still have a logical structure. The development of ideas in this paper suggests, however, that this logical structure for any real case is both complex, subtle, and dynamically changing, even though this structure is built up from a rather small number of simple constructs that serve to define mechanisms such as agreements employed by functional entities such as constrained rational agents. Even worse, the motivating concept — the organization of mind and the mind of organization — is very high level, at the level of metaphor, and is thus made opaque if not invisible by the straw men of rational individualism, with its optimality measures. An informal but illustrative example can perhaps express more intuitively the potential power of the paradigm we are proposing.

One might be inclined to view the selection of a time for a submarine to attack as an optimization problem. The captain of the boat seeks a situation in which the probability he can destroy the surface ship is high enough to dominate the risk of losing his own platform in the attempt. The procedure for each attack will be slightly different depending on several environmental factors (e.g., the position of thermocline layers and convergence zones, wave height, cloud cover, etc.) as well as the active detection attempts being made by the adversary. In order to maximize the boat's chances for survival, each attack scenario could be analyzed and passed over until a favorable condition for attack is attained.

However, construing the attack decision as a static optimization problem obscures the operational complexities inherent in launching an attack as well as the nature of the submarine's dynamic environment. Each attack scenario demands highly coordinated action by the submarine crew because, once the attack is underway, speed and accuracy will be decisive. If the submarine gets off its torpedo and dives before the surface ship's helicopters and P-3C Orions locate and attack it, then it will be safe and the enemy ship must shift it's attention defensive maneuvers. Any delays in launching the torpedo

decrease the probability of success and may lead to the destruction of the sub. The speedy coordination of the various attack tasks will depend on each operational area's capacity to use its information at hand efficiently and resolve conflicts with other functions. Sonar technicians must determine that enemy planes are not nearby. The electronics technicians must feed range information to the weapons officer at the attack center, who must calculate a fire control solution and be prepared to launch. Subsequent diving requires good coordination among planesmen and effective communication from the Executive Officer to the Diving Officer.

The coordination of an attack involves much more than learning one's role by rote. Each process in the attack sequence depends critically on input from other processes, and, at any point, new information may force a change of tactics. Furthermore, each functional officer, based on the requirements of his sub-mission, may have a different opinion about what action should be taken. For example, a sonobouy dropped nearby requires that a decision whether to complete the attack as planned, to maneuver slightly to reduce the probability of detection while continuing the attack sequence, or to descend in order to avoid detection. Completing the attack simplifies the weapons officer's job but exposes the ship to a high probability of detection. Maneuvering reduces the probability of detection but requires that a new fire control solution be calculated and may move the boat to a less advantageous firing position. Diving maximizes the protection of the boat but reduces the number of torpedoes that may be fired or makes the current attack impossible. The weapons officer's desire for a clean shot must be balanced against the sonar officer's instinct to keep the boat hidden.

One could formulate this situation as a new optimization problem, but this would represent only a static snapshot of the operational transitions occurring within the boat in response to the new environment. Suppose instead that an enemy torpedo had been detected. This information would not lead to a change in attack strategy but rather would shift the crew into a wholly different form of coordinated action. (In the language of Schemer, such an event would represent a "critical interrupt.") To be effective in a real battle environment, the crew must be as efficient in changing its structure as it is in running through an attack sequence. In other words, speed and accuracy are not only critical for execution of an attack but also for dynamically adapting plans.

The heart of reflective control suggested by Schemer and the mechanism of agreements is precisely the trade-off between autonomy and interdependence by constrained agents. Reified agreements allow for dependable and fast response to known environments but tend to inhibit change and adaptation. On the other hand, lack of formal agreement may waste resources and make one vulnerable to being divided and conquered. In complicated situations such as the scenario just described, it is easy to see how a Commander could be overwhelmed by the possibilities. If, instead of depending on high level command, each functional officer was provided information about relevant actions by the other officers, appropriate communications protocols, and methods for resolving conflicting goals, the boat might be able to restructure itself and operate much more efficiently in such high tempo situations.[29] It is important to re-emphasize that such agreements may not be modeled statically by defining nodes and information transmit times. Rather, such agreements would work together synthetically within a shared context of dynamic problem-solving in the boat.

---

[29]In some respects, this already occurs. Through experience and training, the crew members learn to internalize the decision framework used by their superiors, which in turn facilitates the development of a terse language for attack coordination and gives the boat some resiliency if direct command is lost.

We do not as yet have exact answers at present to the "how" of managing these trade-offs. However, the research results reported here offer a rigorous starting point for making this question precise. In particular, the computational framework that we have developed has the unique and useful property of being formally defined.[30] In other words, Schemer provides a computational model in which one may precisely describe individual and organizational problem-solving action and interaction. More specifically, we hope to use Schemer to analyze the mechanism of agreements as an extension of the interactions of individuals and groups that operate as constrained rational agents as defined in this computational model.

Most importantly, if the formation and evolution of organizations can be properly elucidated and analyzed using the approach suggested here, then it would seem that the issues of measuring group effectiveness may be resolved, even to the depth that we discussed in the early sections of this document. We fully expect this to be the case. Although much remains to be done, the modeling framework we have developed appears to be adequate for the task. This modeling framework defines the basis for formal modeling as well as defining a computational architecture that can provide a simulation methodology for "experimenting" with alternative approaches to organizational problem-solving. The mechanism of agreements that can be modeled in this framework rests upon a formalizable notion of resource-contrained rational action. Further research on agreements is needed to determine the extent to which the constructs of constrained rationality must be augmented to more fully realize the mechanism of agreements. We envisage these augmentations taking the form of a minimal set of *axioms for social coordination* that provide the added structure for a normative standard for organizational rationality. Only further research will tell.

7. Conclusions.

We have only begun what is likely to be very long-term and rewarding theoretical investigation of organizational rationality. Let us see where we have come and where we may head next.

We first sketched a very general theory of how problem-solvers can act as independent agents and yet come to see why and how they should constitute themselves as an organization. Our ambitious theory of agreements aims to elucidate how a group can be formed by independent agents, the standards of group conduct and group objectives defined, and, once formed, how a group acts rationally to fulfill the group and individual objectives reflected by the founding agreements.

This technical notion of agreements depends crucially on the assumption that independent agents that come to form a group be modeled as *constrained* rational agents. The Schemer computational framework, plus the decision-theoretic methods for allocating and managing problem-solving resources, serve as our basis for making precise our notions of constrained rational agency and multi-agent interactions. Apart from the theory of

---

[30]That is, our approach is certainly unique when compared to other architecture research in AI. Our objective to provide a formalizable basis for architectural features is, to our knowledge, not matched in all this other work. This paper does not focus on Schemer as a formal system. However, in a forthcoming paper we discuss the formal definitional basis of the three basic components of Schemer, the communication model, the formal semantics for the set of control primitives in terms of which all Schemer processes are defined, and the primitive control processes that define the basic Top-Level-Controller.

agreements, this aspect of our research is achieving very valuable results (e.g., Fehling et al., 1989a,b; Fehling & Breese, 1988; D'Ambrosio et al., 1987).

In addition, this Schemer-based research is directly responsible for leading to our more recent interests in organizational problem-solving and organizational effectiveness. Applications of our framework to problem-solving tasks in such domains as advanced aerospace avionics, logistics and resource deployment, automated manufacturing and process management, and command and control have sharpened our concerns about the impact on effective problem-solving of the organizational aspects of these types of systems. In order to prototype and field successful applications in these domains, we have found that Schemer is a practical and conceptually sound approach to modeling these organizational features including many primordial instances of the mechanism of agreements that we have now begun to articulate in this paper.

This paper has focused primarily on the theoretical aspects of this research. However, as we noted in section 2, this work had a very practical beginning. We wanted to formulate a integrated model of command-and-control structures and processes. This model we sought would provide a common language with which to describe the communications, command hierarchy, decision-making processes, mechanical processes, and environment and its processes for a C^2 system. As a result of careful analysis of many small C^2 examples, we believe that we have made a very good start on that task with our Schemer computational architecture.

We expect that the mainstream C^2 research community will view our research as producing at best some new ideas for simulation modeling and testbed development. This will continue to be true until we have more carefully elucidated the formal basis of Schemer and the mechanism of agreements and shown a number of productive analyses of C^2 systems using that basis. An important part of our future research on these problems has exactly that aim. In the meantime, we shall explore the more accessible goal of expanding some Schemer implementation to provide a robust simulation testbed for C^2 methods and applications.

8. Acknowledgements.

We owe our sincere thanks to Dr. "Bill" Vaughan of the Office of Naval Research. He gave us the opportunity to begin looking into this interesting and challenging set of questions about organizational intelligence. He has started us on a long, interesting, and we hope very rewarding journey.

The people who have worked on Schemer and on Schemer-related projects are simply too numerous to list exhaustively. All have contributed to the ideas embodied Schemer in significant ways. Chief among these collaborators are B. Michael Wilber at the Rockwell AI laboratory in Palo Alto and Prof. Bruce D'Ambrosio at Oregon State University. Mike Wilber has been the principle architect and implementer of Schemer for over five years, sheparding this idea through its many stages of evolution. Bruce D'Ambrosio continues to work very closely with us to exploit Schemer's abstractions to obtain useful, new problem-solving methods and theories as well as on many of Schemer's most important applications. Others who have contributed significantly to Schemer's evolution include Michael Shaff and Corrine Ruokangas at Rockwell's Palo Alto Laboratory, and Dr. Stephanie Forrest of the Los Alamos National Laboratory.

Gregg Courand a graduate student of Fehling's at Stanford University is another long-time co-worker, both on Schemer and on the issue of belief management and consensus formation in organizational problem-solving. Gregg's current Ph.D. dissertation is significantly complements the ideas reported in this paper. His discussions with us about these issues have aided us tremendously.

Dr. Jack Breese of Rockwell's Palo Alto Laboratory has worked closely with Fehling to develop a decision-theoretic formulation of methods to manage resource-constrained problem-solving. Jack's technical understanding of decision theory and decision analysis have no doubt improved our use of these notions in the present work.

9. Bibliography.

Abrams, J.M., Chong, C.Y., Fehling, M.R., Rosenschein, J.R., and Tse, E. "Distributed Decision Making Environment." Advanced Information and Decision Systems Final Technical Report, TR-3013-3, April 6, 1984.

Anscombe, G.E.M. *Intention.* (Second Edition). Basil Blackwell, London, 1963.

Baars, B.J. *A cognitive theory of consciousness.* Cambridge, U.K., Cambridge University Press, 1988.

Balzer et al., "Hearsay-III." Information Sciences Technical Report, University of Southern California, 1980.

Barnett, J.A. "How Much is Control Knowledge Worth? A Primitive Example," *Artificial Intelligence,* 22, 1984, pp. 77-89.

Bratman, M. *Intention, Plans, and Practical Reason.* Cambridge, Mass., Harvard University Press, 1987.

Charniak, E. and D. McDermott, *Introduction to Artificial Intelligence*, Reading, Massachusetts, Addison-Wesley,1985.

Cohen, P. R. and E.A. Feigenbaum (eds.) *The Handbook of Artificial Intelligence* (Vol. 3), Los Altos, California, Benjamin Kaufmann, 1982.

Courand, G.J. "Cooperative Problem Solving Via Justification-Based Consensus Formation Processes. Ph.D. Dissertation, Stanford University, in preparation.

D'Ambrosio, B., Fehling, M.R., Forrest, S., Raulefs, P., and Wilber, M. "Real-Time Process-Management for Materials Composition in Chemical Manufacturing," *IEEE-Expert,* Summer, 1987,pp.80-92.

Doran, J.E. and C. Traynor ,"Distributed Planning and Execution- Teamwork 1," Computer Science Technical Report, University of Essex,U.K., 1985.

Drenick, R.F., *A Mathematical Organization Theory*, New York, North Holland, 1986.

Drummond, M. E. "Refining and Extending the Procedural Net," *Proceedings of the 9th IJCAI*, Los Angeles, California, pp. 1010-1012, 1985.

Durfee, E. H. and Lesser, V.R., "Incremental Planning to Control a Blackboard-based Problem Solver," *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1986, 53-64.

Erman, L. and F. Hayes-Roth, V.R. Lesser, & Reddy, D.R. "The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *ACM Computing Surveys* (12:2), 1980.

Fehling, M.R. "Soft Control of Cognitive Processes," In the *Proceedings of 4th Annual Meeting of the Cognitive Science Society*, August, 1982, Ann Arbor, Michigan.

Fehling, M.R. and Erman, L.D. "New Directions for DAI: A Report and Analysis of the 3rd Annual Workshop on Distributed Artificial Intelligence." *ACM SIGART Quarterly*, April, 1983.

Fehling, M.R., Domeshek, E., and Payne, J.R. "An Expert System to Support Fighter IFFN Fusion." Advanced Information & Decision Systems Final Technical Report, TR-1-1036-1, March 1984.

Fehling, M.R. and Lippitz, M.J. Project report on DTDM research at the Palo Alto Laboratory for Intelligent Systems, Rockwell International Science Center, Sept. 1988.

Fehling, M.R., Altman, A., and Wilber, B.M., "The HCVM: An Implementation of Schemer, an Architecture for Reflective, Real-Time Problem-Solving." In Jagganathan,V. and Dodhiawala, R. (Eds.), *Blackboard Systems and Applications*, New York: Academic Press, 1989a.

Fehling, M.R., Einav, D., and Breese, J.S., "Adapted planning and search," *Proceedings of the 1989 AAAI Spring Symposium Series, American Association of Artificial Intelligence*,Stanford, California, March 1989b.

Fehling, M.R. & Wilber, B.M. "Schemer, an Architecture for Distributed, Resouce-Constrained Problem-Solving." Paper in preparation.

Feldbaum, A.A. "Dual-Control Theory I-IV," *Optimal Control Systems*, New York, Academic Press, 1965.

Genesereth, M. R. and Smith, D. E., "Meta-Level Architecture," Technical Report HPP-81-6, Stanford University, 1982.

Georgeff, M., Lansky, A., and Schoppers, M., "Reasoning and Planning in Dynamic Domains: An Experiment with a Mobile Robot," SRI International Technical Note 380, SRI International, Menlo PArk, CA, April, 1987.

Goode, I.J. *Good Thinking: The Foundations of Probability and Its Applications*. Minneapolis, University of Minnesota Press, 1983.

Guffey, J. "AI Takes Off: Expert Systems that are Solving In-Flight Avionics Problems,"*Aviation Week and Space Technology*, Feb. 16, 1986, pp.

Hayes-Roth, B. " A Blackboard Architecture for Control" *Artificial Intelligence* (26:3), 1985, pp. 251-321.

Heidegger, M. *Being and Time*. Translated by John Macquarrie and Edward Robinson. New York, Harper and Row, 1963.

Howard, R.A.,"The Foundations of Decision Analysis," *IEEE Transactions on Systems Science, and Cybernetics* SSC-4:211-219, 1968.

Keller, K. & Bedoya, C. "An Expert System Application to Maintenance of Inertial Navigation Systems," In the *Proceeding of the 1983 NAECON Conference.*

Kim, K.H. and Roush, F.W. *Team Theory*. Chichester, U.K., Ellis Horwood Limited, 1987.

Lesser, V.R., Pavlin, J., and Durfee, E., "Approximate Processing in Real-Time Problem Solving," *AI Magazine*, 9:1, 1988, pp.49-61.

Minsky, M. *The Society of Mind*. New York, Simon and Schuster, 1986.

Peterson, J.L. *Petri Net Theory and the Modeling of Systems*. Prentice Hall Inc., Englewood Cliffs, New Jersey, 1981.

Pfeffer, J., "Organizations and Organization Theory," in Lindzey and Aronson (eds.), *The Handbook of Social Psychology (3rd Edition)*, Reading, Mass., Addison-Wesley, 1985.

Raiffa, H. *Decision Analysis: Introductory Lectures on Choices under Uncertainty*. New York, Random House, 1968.

Ramsey, F.P. *The Foundations of Mathematics*. Totowa, New Jersey, Littlefield, Adams, & Co., 1965.

Raulefs, P., D'Ambrosio, B., Fehling, M., Forrest, S., and Wilber, M. Knowledge-based control of a continuous material processing system. In *Proceedings of the 3rd IEEE AI Applications Conference*, Feb. 1987, Orlando, FL.

Reid, L.G. "A model of communication and control in computing." Ph.D. Thesis, Carnegie-Mellon University, 1980.

Rosenschein, J.S. and Singh, V. "The Utility of Meta-Level Effort," Heuristic Programming Project, Report No. HPP-83-20, March, 1983

Savage, L. J., *The Foundation of Statistics*., New York, Dover: Press, 1972 (reprint of original published edition, 1954).

Simon. H.A. *Sciences of the Artificial*. Cambridge, Mass., MIT Press, 1969.

Tate, A., "Project Planning using a Hierarchical Non-Linear Planner," Department of Artificial Intelligence Report 25, University of Edinburgh, 1976.

Tate, A., "A review of Knowledge Based Planning Techniques," Knowledge Engineering Review, 1986, pp. 4-17, 1986.

Tate, A. and A. M. Whiter," Planning with Multiple Resource Constraints and an Application to a Naval Planning Problem," *Proceedings of the 1st Conference on Applications of Artificial Intelligence*, Denver, Co. , 1984 .

Tse, E. & Bar-Shalom, Y. "Actively Adaptive Control for Nonlinear Stochastic Systems," *Proceedings of the IEEE*, vol. 64, no. 8, August 1976, pp. 1172-1181.

Van Creveld, M., *Command in War*, Cambridge, Mass., Harvard Press, 1985.

Vere, S. "Planning in Time: Windows and Durations for Activities and Goals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5, No.3, 1983, pp. 246-267.

von Winterfeldt, D. and Edwards, W. *Decision Analysis and Behavioral Decision Theory.* Cambridge, U.K., Cambridge University Press, 1986.

Warren, D.H.D. "Generating Conditional Plans and Programs," *Proceedings of the AISB Summer Conference*, Edinburgh, 1976, pp. 344-354.

Winograd, T. and Flores, F. *Understanding Computers and Cognition.* Norwood N.J., Ablex Publishing Corp., 1986.